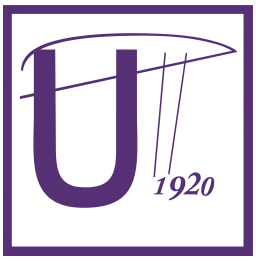


Space-Time Window Reconstruction in Parallel High Performance Numeric Simulations. Application for CFD

Anton Alin-Adrian

Scientific supervisor: Prof.Dr.Eng. Creţu Vladimir-Ioan

Thesis submitted for the fulfilment of the requirements
for the degree of Doctor of Philosophy

28^{th} of November 2011

Faculty of Automation and Computers
"Politehnica" University of Timişoara

Dedicated to my uncle, who has hallmarked me
into the way of science, and peacefully passed away...
To my family, with love.

Cuvânt înainte

M-am întâlnit cu calculatorul şi tehnica de calcul înainte de începerea şcolii, iar cu Internet-ul încă de la începuturile sale. A devenit o pasiune pe care mi-am cultivat-o apoi în anii de şcoală, dar nu mi-am închipuit că o voi aprofunda şi printr-un doctorat.

Să te apuci să faci un doctorat imediat după finalizarea studiilor, chiar dacă ai trecut şi printr-un masterat, e dificil şi necesită o anumită doză de inconştienţă. În astfel de momente din viaţă e important să ai lângă tine personalităţi şi modele autentice, care pot să cântărească şi să-ţi arate cu delicateţea necesară drumul pe care ai putea să-l parcurgi.

Am avut cu siguranţă şansa, ca în astfel de momente decisive, să am alături de mine o personalitate inestimabilă – unchiul meu – care mi-a deschis gustul pentru performanţă şi mi-a arătat drumul spre ea. Din păcate s-a hotărât să revină din această viaţă, cu deplină seninătate, la începutul acestui an, când înfloresc pomii înainte să dea în fruct.

Am fost zguduit şi dezamăgit, pentru că mai aveam atâtea de discutat. Am luat atunci hotărârea să-i dedic în întregime teza.

Pe domnul Profesor Creţu l-am cunoscut de când eram copil. Am avut privilegiul pe tot parcursul colaborării noastre, să beneficiez de profesionalismul şi caracterul său excepţional, de perseverenţa sa în modestie şi răbdare. Domnul Profesor Creţu Vladimir împleteşte cu modestie abilităţile profesionale cu virtuţile personale, şi îi voi fi recunoscător întotdeauna pentru purtarea de grijă şi amprenta pe care şi-a lăsat-o în viaţa mea, ca autentic şi demn exemplu.

Cercetătorul Muntean Sebastian a fost alături de mine în toate momentele cheie ale carierei mele în devenire, ne leagă o prietenie sinceră, şi îi sunt recunoscător pentru efortul acordat de-a lungul timpului – fiind practic după domnul Profesor Creţu cel mai apropiat colaborator în problemele ridicate la doctorat.

Le mulţumesc din suflet tuturor dascălilor mei, şi îi asigur pe această cale că fiecare strop de învăţătură, venit din exemplul personal şi pasiune, s-a strecurat prin timp şi a devenit o candelă eternă, pe care o aprind când simt nevoia să mă uit la cer. Îmi amintesc cu drag când cel dintâi pe care l-am întâlnit, în persoana d-nei învăţătoare, a rămas marcat de textul scris pe uşa camerei mele, sus, cu litere strâmbe: "omul de ştiinţă Adrian". Nu lipsea nici una, deşi am deprins acest lucru mult mai târziu la limba română, la scurt timp după ce m-am împrietenit şi cu fizica. Adresez profundă recunoştinţă celor care m-au ajutat şi au investit în construcţia mea.

În sfârşit, doresc să mulţumesc părinţilor mei şi familiei, fără a căror susţinere, răbdare şi educaţie nimic nu ar fi fost posibil. Sprijinul intelectual,

emoţional şi financiar pe care l-am primit obligă, şi nu poate fi răsplătit. Este poate şi motivul pentru care am încercat să dau tot ce pot, în orice fel de circumstanţe.

Nădăjduiesc ca numele Politehnicii din Timişoara să rămână în continuare, la aproape un centenar de la înfiinţarea ei, şi la jumătate de veac de la construcţia primului calculator românesc din mediul universitar, o emblemă de calitate şi seriozitate pentru viitoarele generaţii, purtând cât mai sus povara blazonului cu care a fost învestită de către autorităţile în drept, la ctitorirea ei.

Sper să mă alătur prin această lucrare generaţiilor de studenţi şi dascăli care au adus o fărâmă de strălucire acestui blazon.

Autorul

Abstract

The size of the output originating from large scale, numerical simulations poses major bottlenecks in high performance, parallel computing. Recently it became more and more evident that a radical change has to take place in the way scientists and engineers handle numerical simulations. The beating up of more computational horse power out of supercomputers, is a trend that simply hits the data wall long before it gets a chance to start the ExaFLOP race. Supercomputing today is like riding a barouche with horses that travel orders of magnitude faster than the storage; long distance runs, add hills, and valleys, to the landscape; high performance computing facilities, have become high-tech aquaria, where one can build the most advanced, and expensive submarines, and then be limited to only staring at them through the windows.

This thesis proposes a new concept for dealing with large-scale, numerical simulation data. The new concept is called 'space-time window reconstruction', and introduces a new style in high performance computing.

A space-time window is an independent numerical simulation, based on a large scale version, capturing a subdomain of analysis, in both time and space.

The concept is implemented using two different solutions: the first is focused on providing maximum flexibility to the user, while still retaining the flow features from the global simulation; the second concentrates in reconstructing the very same floating point bits. Which one is used depends on the user. Both provide substantial data reduction, and alleviate the supercomputing data bottlenecks. However, they are more powerful when used together, in a compact, stand-alone procedure.

A regular computational fluid dynamics, numerical simulation, is always shifted in time, and sometimes even in space. It is common practice to analyse a domain larger than what is really of interest, just to get the experimental data to match against a smaller region in the space-time domain. It is thus natural to propose a style of analysis that tries to extract and decouple the interesting simulation parts, from the large scale version, which is normally tightened to expensive, and scarce hardware installations. The obtained results lay down the foundations for a new way of doing numerical analysis, which can be extended to many other scientific fields, like, for instance, electrotechnics and magnetohydrodynamics. The overall data reduction varies from 2× to 6% or better, but the proposed methods can only be applied with certain restrictions in place.

In the closure, the thesis outlines a number of research directions which could be approached in the future, and retains a firm position that the new paradigm has to be polished by joining forces with colleagues from applied

mathematics and with experts in computational fluid dynamics (CFD). A list with claims and personal contributions is also explained.

Keywords: data reduction, space-time window, numerical simulation, CFD

Acknowledgements

I am grateful and thank my PhD supervisor Prof. Dr. Eng. Creţu Vladimir-Ioan, for all of his support, help, patience, strength of character and guidance. Obsessed with the highest level of quality for scientific research, I am deeply indebted to Dr. Eng. Muntean Sebastian, for his most genuine help and advices.

Also, I acknowledge, credit and thank Prof. Dr. Eng. Resiga Romeo and Dr. Eng. Ruprecht Albert for research directions, for their solid support and expertise, and for allowing me to work in first hand research teams.

Next, my best thoughts go to Ms. Ivana Buntic Ogor, who prepared the square cylinder test case for me, based on the work of Dr. Niklas Nordin. Also, I am grateful to Dr. Bistrian Diana Alina who has supported me with mathematical details.

Numerical analysis and simulation is a difficult topic. The research teams from CNISFC in Timişoara, and IHS in Stuttgart, have been overwhelmingly supportive, and we would like to thank them for their assistance.

On the other hand, the project has received full support from the Department of Computer and Software Engineering, in terms of logistics, expertise, documentation, and glorious paperwork victories. Namely, I gratefully thank all my colleagues for their sustainment, and my office mates, from both Timişoara and Stuttgart, for all the long winters, encouragements, keen support, hot summers, and the things that are better left unsaid.

This work was partially supported by the strategic grant POSDRU 6/1.5/S/13 (2008, project id 6998) of the Ministry of Labour, Family and Social Protection, Romania, co-financed by the European Social Fund – Investing in People. The research would not have been possible without the financial efforts sustained by the management of the University, and the steadfastness of the rectorate.

I gratefully thank the bwGRiD project [1] for the computational resources.

If by any chance you think there's something missing and that you have been mistakenly left out, then I must have forgotten to include your name in the sources.

Please fill your name in here, beginning with capital letters:
. . . . . .
and you are gratefully acknowledged.

Contents

List of Figures

List of Tables

Abbreviations

ACM Association for Computing Machinery 7

AI artificial intelligence 22

ASIC application-specific integrated circuit 28

CAD computer aided design 39

CAS computer aided surgery 21

CFD computational fluid dynamics vi

CiSE computing in science and engineering 9

CNISFC Centrul Naţional pentru Ingineria Sistemelor cu Fluide Complexe

CSE computational science and engineering 1

CT computer tomography 21

DEISA Distributed European Infrastructure for Supercomputing Applications 11

DNS direct numerical simulation 29

DSP digital signal processing 29

DWT discrete wavelet transform 29

ECC error correcting code 140

FEM finite element method 5

FVM finite volume method 5

FLOPS floating point operations per second 11

FP floating point 26

FPGA field-programmable gate array 28

GPU Graphics Processing Unit 28

GPGPU General-Purpose computation on the Graphics Processing Unit 97

GRIB GRIdded Binary 23

GROWTH German-Romanian Workshop on Turbomachinery Hydrodynamics 3

GTS Global Telecommunication System 23

HET HPC European Taskforce 11

HLRS Höchstleistungsrechenzentrum Stuttgart 157

HPC High Performance Computing 4

IEEE Institute of Electrical and Electronics Engineers

IEEE-CS Institute of Electrical and Electronics Engineers – Computer Society 7

IHS Institut für Strömungsmechanik und Hydraulische Strömungsmaschinen 157

ILP Instruction Level Parallelism 26

LAN local area network 24

LES large eddy simulation 117

MPP massively parallel processor 17

NWP numerical weather prediction 23

ODE ordinary differential equation 26

PDE partial differential equation 3

PE processing element 18

PRACE Partnership for Advanced Computing in Europe 11

RAM Random Access Memory 140

RLE run length encoding 26

RPM rotations per minute 10

SMP symmetric multi-processor 18

SSDB scientific and statistical databases 26

VR virtual reality 21

‘A good soldier is a poor scout’.

Cheyenne

1 Introduction

This chapter introduces the reader to one of the main challenges for computational science and engineering (CSE). The motivation for this thesis is driven by the need to have robust methods that can deal with very large numerical simulation data. This is needed both for storage and transportation. High performance numerical simulation bottlenecks are given by different states of the digital information. Data originates from computation, proceeds throughout the communication networks and finally takes physical form in storage devices. In reality, the process is much more interleaved. The problem is that computational power has increased much faster than the ability to handle the output. Therefore, supercomputing today is like riding a barouche with horses that travel orders of magnitude faster than the storage. Long distance runs add hills and valleys to the landscape. New ways of thinking must emerge in order to tackle with the problems already at hand. It is not very practical to produce a once in a lifetime, most polished numerical simulation, if then one is dead in the water with it. In this respect, supercomputer facilities of our days are like high-tech aquaria where one can build the most advanced and expensive submarines, and then be limited to only staring at them through the windows. The whole picture is thoroughly explained in the following section; afterwards, the closure outlines the structure of the thesis.

1.1 Motivation

This section outlines the main motivations behind the thesis research. The first generation of computers are capable of cycling around 50-70 operations per second. Half a century ago, a computer conceived and built in Timişoara has been used for designing the hydroelectric dam at Vidraru, on the Argeş river [11].

Present day computers perform 10^{15} times faster [12], and it is projected that the exa-FLOPS barrier will be breached between 2020-2025. However,

our ability to store and transport information of this magnitude to consumer hardware, for scientists and engineers to interpret, is lagging behind [13].

Many attempts for dealing with these problems are still made. The people from the applied mathematics field have been developing ways for reducing numerical simulation data to the minimum size. The majority of these methods have been implemented in structural analysis domains, and are known, in a more generic manner, as part of the global-local analysis framework[14, 15].

On the other hand, the people from computer science fields are also developing methods of their very own. Floating point compression algorithms, for example, based on prediction [7] and heuristic methods [16], wavelet solutions [17], peer to peer data transport [6], and massive data infrastructures like in [18], are to name just a few. The number of attack angles for dealing with large numerical simulation data is vast. Numerous approaches from the literature are thoroughly examined in the following chapters.

It is only natural that the end-users, who endorse in large scale numerical analysis, like CFD scientists, are handling the problems in their own way. The easiest and most frequent workaround is to subsample the results until something that can be downloaded and post-processed at hand can be obtained. Skipping some of the time steps really seems to conform to this Ockham's razor approach, however, it may create problems for space-time visualisation and post-processing.

In situ post-processing, on the other hand, is a more demanding method, and enables the user to process all the data remotely. A very good survey in this respect is given in [19]. Feature tracking and extraction allow the scientist to focus and only deal with essential phenomenological information [20].

Different schools, different approaches, and only one problem to tackle with: a robust, feasible way, for dealing with large numerical simulation results. Unlike the state of the art solutions, which are shown to only scratch the surface of the real problem, this thesis introduces a new concept, called 'space-time window reconstruction'. The idea brings a paradigm shift in numerical simulations, with benefits for the scientific community, and direct contributions in CFD. The next section introduces the main objectives for the research.

1.2 Thesis objectives

This section explains the objectives that had been established for the thesis. Beginning with the 1960s, the people in aerospace engineering have

been developing and using private tricks of the trade to carry on with numerical analysis, on poor hardware. The limitations in computer memory and performance determined the users to invent workarounds. Sometimes these workarounds bear different names for the same methods, depending on geographical location and the company involved. Most of the knowledge remained black art, surfacing later in different numerical analysis fields.

The target of this thesis is to find robust ways to minimise the size of the data produced by large scale, parallel numerical simulations. Such simulations are most likely modelling unsteady phenomena (using real time as a component). Since numerical analysis has plenty of applications, this thesis is focused on CFD.

With CFD in mind, one needs to understand why techniques like submodelling [15] or substructuring [14] are not borrowed by the community, after half a century of existing practice. There must be a reason for which transient implementations are scarce, and the common applications of these methods involve only static analysis. To answer these questions, a new concept is proposed.

During the 5^{th} German-Romanian Workshop on Turbomachinery Hydrodynamics (GROWTH), held in July 2009 in Timişoara, the seeds for what it later became known as the 'space-time window reconstruction concept' have sparkled. This concept is based on the fact that often enough, a CFD user has to produce simulations for very large analysis domains, while only a small portion of the results, in both time and space, are of interest. The main objective of this thesis is to produce the 'space-time window reconstruction' concept, and use it as a means for bottleneck alleviation in large scale, parallel HPC numerical simulations.

Numerical analysis and domain decomposition are mathematical subfields with a very broad spectrum of interdisciplinarity. Numerous scientific and engineering schools are involved, but with very little coupling along the communities. Anyone who has ever dared to peek inside numerical analysis software, commercial or not, recognises the esoteric programming style – owed to the lack of communication between software engineers, on one hand, and mathematicians who understand applied mathematics, on the other. The goal is to integrate what's best available in all of the relevant scientific branches, and find a 'data minimising' solution for CFD applications.

Modern numerical analysis, including CFD, can not exist without computers. In fact, history shows that most of the machines and devices designed for computing had some sort of scientific or mathematical motivation in the background [21, 22]. Either for solving partial differential equation (PDE) systems [23], projectile trajectories [24] or cipher breaking [25], computers had quickly become the third pillar in science – alongside theoretical and

experimental research.

Since this is a computer science thesis, the focus is on data size reduction, with benefits in storage and transportation. The next section outlines the organisation of the document.

1.3 Thesis outline

This section describes the structure of the thesis; the rest of this document is organised as follows:

Chapter 2 describes the background in the field of, and the strategy applied for, understanding High Performance Computing (HPC). There are several publications covering this chapter. The unconventional approaches to the literature fortunately pay off, and a new perspective for looking at computational science and engineering (CSE) is introduced, from the computational standpoint.

Chapter 3 contains a thorough survey of the state of the art in dealing with large numerical simulation data. Different, independent schools of thought are drilled for information, with a synthesis of the most relevant solutions from computer science, applied mathematics and computational fluid dynamics. The conclusion is centred around the observation that none of the methods from the state of the art have enough interdisciplinarity to grasp the problem at its own roots. Therefore, they only scratch the surface of it and fail to provide an effective solution.

Chapter 4 proposes the 'space-time window reconstruction' concept, provided with two solutions for implementation. The two remedies for the problems in the state of the art can be used either independently of one another, or with combined force for maximum efficiency. The first one is based on submodelling, and provides maximum flexibility, and the second one is a refined validation of solution A, where the interpolation is completely removed, and the internal fields inside the space-time window can be reconstructed with bit-level accuracy. It is based on the interception of parallel interprocessor traffic.

Chapter 5 introduces the strategy applied for trying to tackle with the aforementioned problems, and therefore validating the new concept, by dealing with test cases of gradually increasing complexity. The strategy is designed with fail-back plans, because each of the test cases emerge from complete uncertainty as foundations for the next level. Plan B corresponds to solution B, and it is fortunately studied at the end, after all the other stages have proven to be successful.

The first test case, in Chapter 6, is a proof of concept for submodelling in

CFD. It is a trivial simulation using the finite element method (FEM), but like any first step in uncharted territories, its first purpose is to encourage the research in the planned direction, and to provide basic information about the problems that need to be considered for the more complex targets.

The next one, in Chapter 7, is meant to verify if the procedure holds for rotationary flows. Also, it is designed as a transition from the finite element method (FEM) to the finite volume method (FVM), and as a switch from the old PETSc toolkit[26] to the OpenFOAM[27].

Chapter 8 shows a full 3D unsteady demonstration of submodelling, basically introducing the concept of complete space-time window reconstruction in CFD. This method is aimed for users who desire to obtain flexibility, while still reducing the size of the data, and preserving the flow features.

Chapter 9 presents a different solution for space-time window reconstruction, based on the interception of parallel interprocessor traffic. The method is designed for users who want to trade flexibility for perfectly accurate reconstructions.

The concluding chapter synthesizes the contributions, thoroughly draws the final remarks, and outlines the future research directions. As a fundamental conclusion, a compact procedure for resolving the numerical data deluge problem is presented, based on the two solutions proposed in Chapter 4. This enables the user to bring the most important parts of the large scale simulation on to commodity hardware, like laptops and memory cards, and perform local post-processing with unprecedented flexibility and freedom.

The thesis now continues with the chapter on High Performance Computing (HPC).

2

High performance computing

2.1 Research strategy for understanding high performance computing

Before continuing with the thesis, it is important to underline the background in High Performance Computing. This section presents the strategy used by the author to understand HPC, and the original viewpoint obtained during the first year of study.

During the diploma thesis, key problems for store-and-forward communication systems had been addressed, with a very specific application for mail transfer agents. The contributions had been strictly computer science oriented, with improvements for operating system services. The first contact with CFD has taken place during the master's program [21]. The final year of the master's program was spent trying to get the grasp of what HPC is, and how it relates to the problems that need to be addressed by CFD – in particular – and numerical analysis, in general. Meanwhile, the origins of computational devices have been studied, and the key historic architectures have been organised into a fresh perspective. The author concluded that scientific computing and numerical analysis were the main driving forces for the development of modern supercomputers.

In order to cut throughout the marketing fog that surrounds the supercomputer business, data mining techniques have been applied, on a collection of about 6000 related scientific articles [28]. Both partitioning around medoids [29] – Fig. 2.2 – and single linkage hierarchical clustering [30] – Fig. 2.1 – have been used, with comparable results.

Most of the papers have been selected from conference proceedings and journals published by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers – Computer Society (IEEE-CS). Based on silhouette graphs, it was concluded that the relationships between the studied publications were reasonable, but artificial

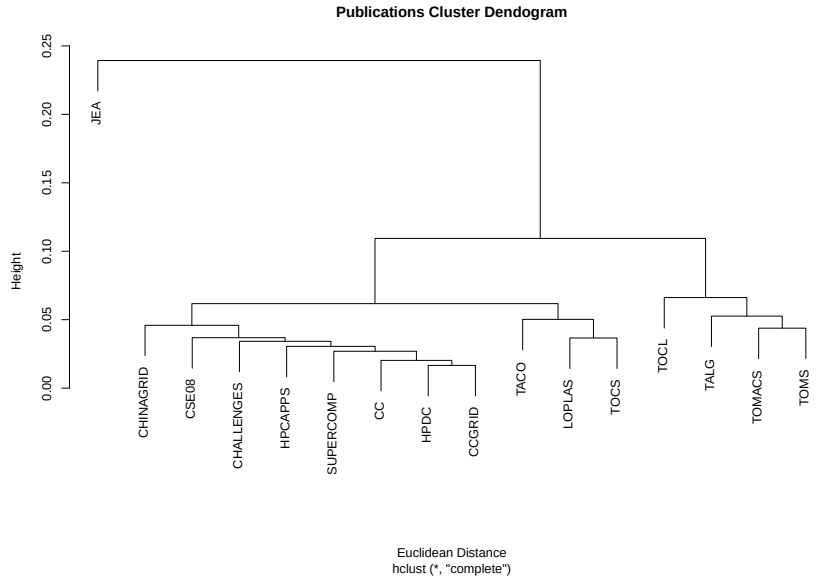


Figure 2.1: Hierarchical Clustering

[31]. However, since the study has been based on regular term frequencies rather than more appropriate inverse term metrics, the relationships between the periodicals merely reflect the writing style of the articles, not the content.

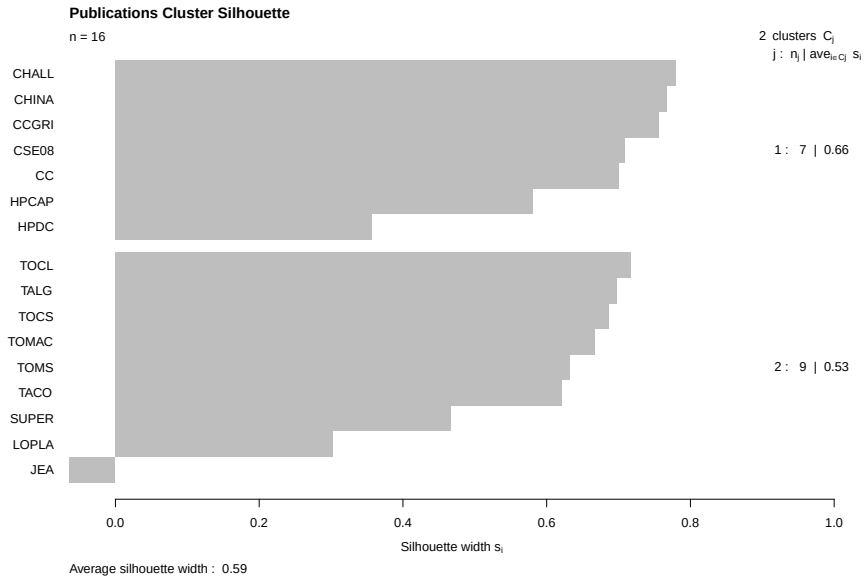


Figure 2.2: Partitioning Around Medoids

In defiance of that, the data acquisition process provided the blueprints

for grasping computing in science and engineering (CiSE), and for developing an original model of the subject, detailed in [32] – see Fig. 2.3.

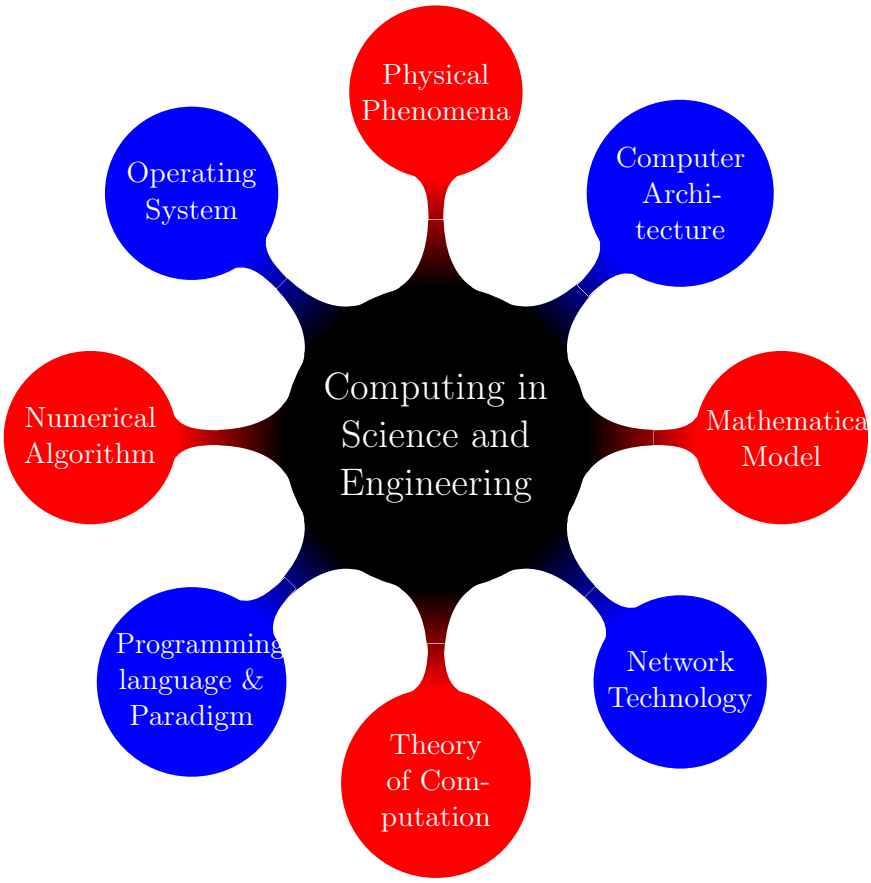


Figure 2.3: Computing in Science and Engineering

Armed with the new knowledge, it has been easier to investigate the literature for specific issues. Computer science is governed by two main international societies, the ACM and the IEEE-CS. These two institutions are powerful enough to prescribe the computing curricula around the world, and help shape the trends and the future of the field.

The ACM for instance, publishes well-targeted surveys on computer science. In Sameh et al. [33] the author specifically prospects the emergence of CSE, and lays down the foundation for anyone trying to understand the confluence between computers and modern science. More general purpose periodicals, like the 'Computer' magazine and the 'Transactions on Computers', published by the IEEE-CS, along with the 'Communications of the ACM', can also provide starting points for investigations; the IEEE 'Spectrum' magazine has a computer section, with useful information.

On the other hand, the literature that deals with CSE from a computational perspective is very rare. In 1999 the IEEE-CS started to print the first CSE magazine, which has been targeting computer science audiences ever since [34]. The 'International Journal of High Performance Computing Applications', published by SAGE Ltd., and the 'Journal of Supercomputing', published by Springer Science+Business Media, also keep an open computational perspective. The ACM 'Transactions on Mathematical Software' and the 'Transactions on Modelling and Computer Simulation' have more specific targets, merely numerical algorithms, and non-numerical simulations, respectively.

Tveito and Winther [35] introduce the reader to the methods that are used when solving partial differential equations with computers. Together with the easy to follow implementations of Nikishkov [36], the two books provide computer scientists with common CSE foundations. Several other materials can provide background state of the art information, like Tucker [37], Bader [38] and Benjamin [39].

Modern numerical analysis is considered to begin with the 1947 paper, by John von Neumann and Herman Goldstine, 'Numerical Inverting of Matrices of High Order' [40]. The well known bible of scientific computing is Press et al. [41], a masterpiece in applied mathematics; also, a recent revision of Bird et al. [42] explains the fundamental equations on transport phenomena.

Numerical analysis has driven the development of modern computers, together with the opportunity, and necessity, to solve larger and more complex mathematical problems, in science and engineering. The following section presents a short synthesis on the evolution of modern supercomputing, up to the present and the future.

2.2 Evolution of supercomputing

This section outlines the evolution of supercomputers since the beginning of computational hardware, up to the present, and continuing with prospects for the future.

The very first computers were designed to solve mathematical and scientific problems. The Z1 model of Konrad Zuse, from 1938, was the world's first freely programmable, Turing-complete computer [43], using Boolean logic and binary floating point numbers [44]. The Atanasoff-Berry from 1939 was specifically designed to solve linear systems of equations; it was not programmable [23]. Many others followed, paving the road for modern CSE.

The earliest machines used a clock cycle of 1 Hz, limited by the number of rotations per minute (RPM) of various components. Modern computers

use different metrics for understanding performance. In terms of numerical and scientific computing, the number of floating point operations per second (FLOPS) is the most relevant. Starting with a series of papers from 1983 to 1992, Jack Dongarra introduced the idea of benchmarking computers with the use of linear equation solvers [45–49]. The first LINPACK report dates back to 1979 [50].

The problem with FLOPS is that they fail to capture the real performance of the machine. They can reflect the peak level, when the system is benchmarked with specially crafted tests, the theoretical level only achieved by pencil and paper, and the real, averaged level, when the machine is running genuine scientific applications, from the default workload.

The real performance is always lower than both the peak and the theoretical levels, and highly depends on the nature of the application which is in use. Dongarra proposed a more complex set of benchmarks for supercomputers, which also stresses the bandwidth bottlenecks from the internal system, and thus provide more realistic values for the metrics [51].

The people from the Distributed European Infrastructure for Supercomputing Applications (DEISA) and the Partnership for Advanced Computing in Europe (PRACE) projects make use of more relevant metrics for benchmarking their systems. Two separate benchmark suites containing different software applications, covering the broad spectrum of computational sciences, are carefully maintained. For CFD, FENFLOSS is used – a code that has been developed for more than 20 years at the 'Institut für Strömungsmechanik und Hydraulische Strömungsmaschinen', University of Stuttgart [52].

The roadmap to European petascale computing is depicted in Fig. 2.4. The HPCEUR project that started in 2004 morphed into the HPC European Taskforce (HET), bringing scientists together for the creation of a scientific case.

The PRACE initiative is aimed to link Tier-0 country-level European supercomputing sites. The Memorandum of Understanding (MoU) was signed on the 16^{th} of April 2007 by 16 member states in Berlin. The DEISA projects are lower level, Tier-1 and Tier-2 connections between regional and local HPC centres, with dedicated European backbones, to allow for local sites to benefit from cutting-edge, supercomputing facilities.

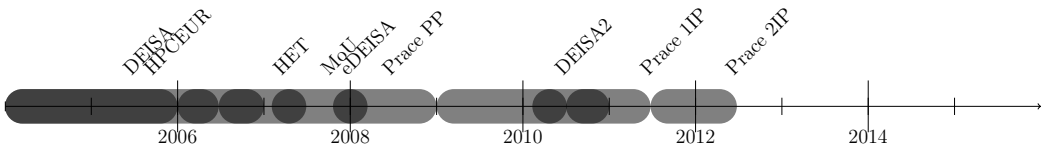


Figure 2.4: The European Road to Petascale Computing

The combined PRACE/DEISA ecosystem is designed to provide the EU with a living HPC organism, having highways and neural centres, in consistency with the European target to become the number one engine in scientific innovation. A global filesystem is already accessible via grid middleware through the GEANT backbone. Expensive FP6 and FP7 projects had already been consumed.

The two projects have naturally merged, with 20 EU and associated countries, and cutting-edge supercomputing centres, marching together for European Petascale access. The GEANT-supplied private backbone is continuously extended to connect as many academic sites as possible. The PRACE project is now in the second implementation phase (2IP), until the middle of 2012.

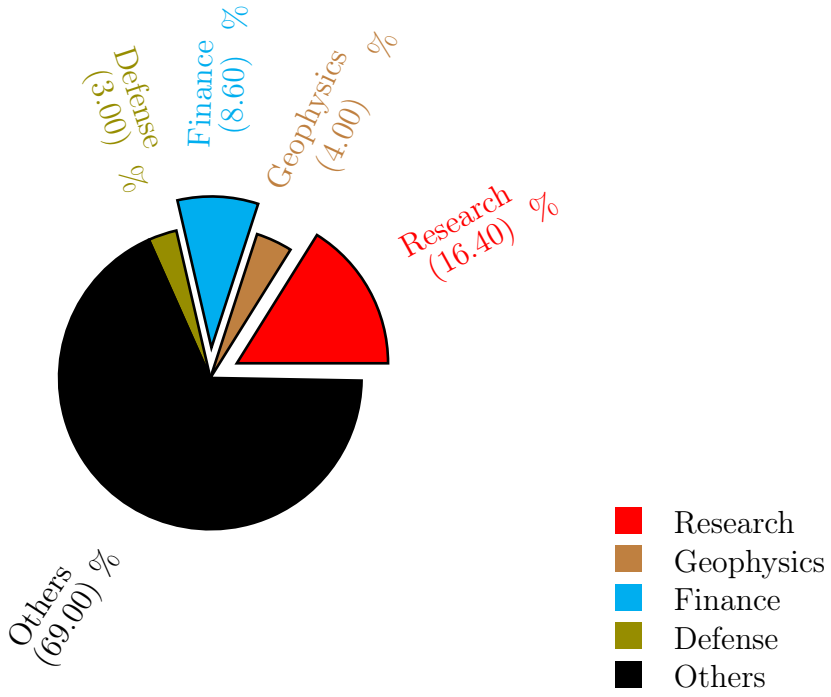


Figure 2.5: World-Wide Supercomputer Applications

Fig. 2.5 shows the most prominent applications of supercomputers throughout the world. Research and finance are leading domains. Many of the participants in the list do not specify their applications; however Fig. 2.6 considers a more detailed presentation.

Besides research and finance, logistic services, geophysics, defence and information processing share the largest number of HPC sites. In spite of the lower ranks associated with the rest of the domains, the last position in the

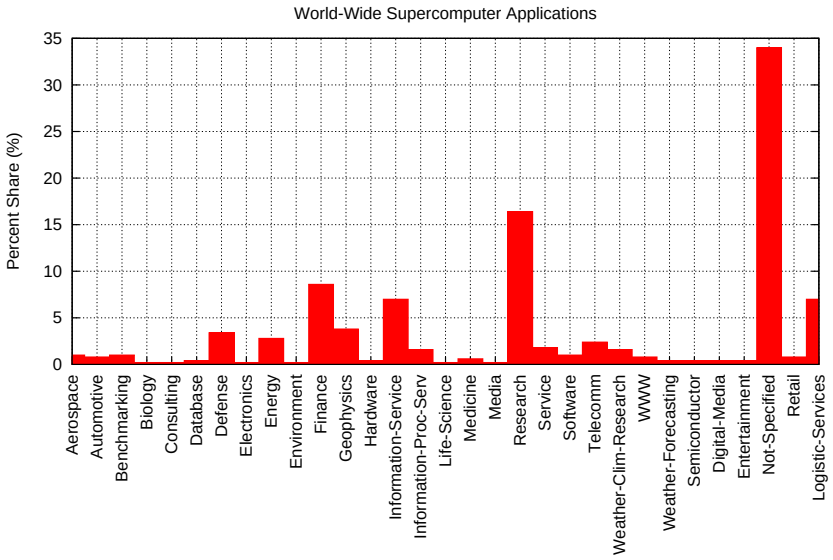


Figure 2.6: Supercomputer Applications

list is already a TFLOP/s machine.

There are many other engineering and scientific fields that benefit from high performance computing. Almost every aspect of scientific research has been improved with the help of computers – from historical and social humanistic studies, to particle physics, protein folding, and computational chemistry. Whenever the specificity of the problem allows for loosely-coupled systems to be used, quasi-supercomputing systems can be constructed by volunteer computing projects. As of March 2011, the cumulated power of the BOINC network [53] is that of two Tianhe-1A systems which rank 1 in the Top500 list. Some of the projects also exploit GPU accelerators to speed-up the 'number crunching'.

In terms of the number of supercomputing sites, Europe shares one quarter of the world-wide systems – Fig. 2.7. The American continent clearly dominates the market. With China beginning to produce its own microprocessors, and the Tianhe system knocking off the first rank, the Asian continent is promising to catch-up. The same goes for the volunteering desktop computing projects; according to the BOINC statistics, the US is followed by Germany, UK, Canada and Japan.

Again, it is important to remember that statistics have always been used for marketing purposes. The number of FLOPS a system can perform have become more or less irrelevant with time. During the age of third-generation computers the software is of much more importance, and it is critical to

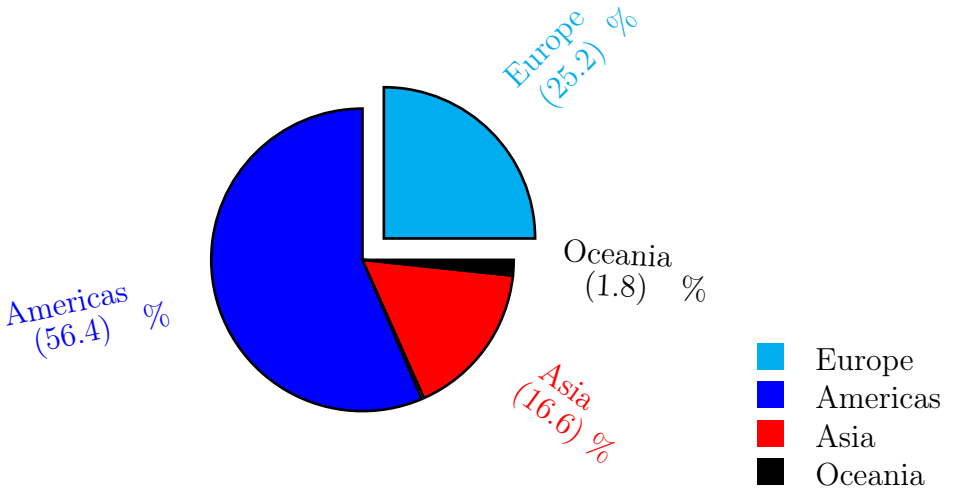


Figure 2.7: Continental Supercomputer Systems

identify the system which will perform better when running very specific software implementations. Nonetheless, the LINPACK Benchmark is widely used to solve a dense system of linear equations on the world's supercomputers, and produce the Top500 list [12]. The portable implementation uses 64-bit floating point operations and can be scaled to various system architectures.

Fig. 2.8 shows the evolution of performance using historical supercomputers, considered to be the top of the edge achievements during their time. It is based on the work of Dongarra et al. [2].

The Y-axis is logarithmic. From Z1 to the RoadRunner (RR), the evolution appears to be predictable. According to Fig. 2.8, the ExaFLOP – 10^{18} – barrier will be broken around 2022 or in any case between 2020 and 2025.

Feitelson even proposed a formula for estimating the future performance of Top500 computers [54]. It is listed in equation (2.1).

$$Rmax(r,t) = Rmax(r_0,t_0)(\frac{r_0}{r})^{0.7}2^{\frac{(t-t_0)}{1.15}} \tag{2.1}$$

In (2.1) r is the rank in the list and t the year, and r_0, t_0 are the current indexes; Rmax is the maximal achieved performance expressed in GFLOPS, and can be taken from the list. According to the prediction formula, the ExaFLOP barrier will be taken down in 2020, and in 2022 the rank 1 position will triple the speed. The estimation has been based on the Tianhe-1A leadership.

Fig. 2.9 shows the performance growth of the supercomputers using the TOP500. It is based on Strohmaier and Meuer [3] where the authors correctly

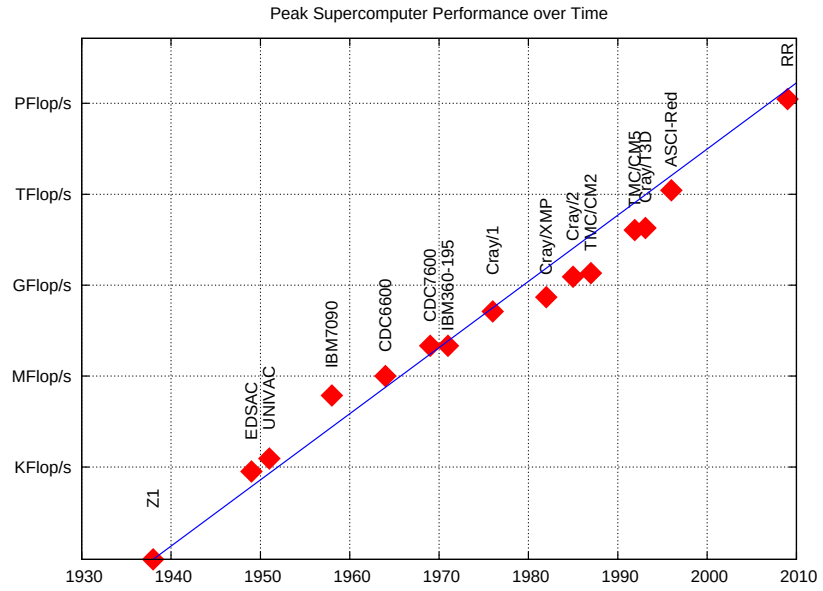


Figure 2.8: Historical Peak Performance – based on Dongarra et al. [2]

extrapolated that the PetaFLOP limit will be breached at the end of 2008, six years before it happened.

The lowest rank clearly shows a uniform, exponential distribution appearing as a straight line on the semi-log axes, but the first position in the list is jumpy. The total cumulated power of the 500 ranks is suggesting that the world-wide supercomputer performance is indeed experiencing an exponential development. This is in agreement with Fig. 2.8, where Z1 and RoadRunner (RR) are on the same track.

Fig. 2.10 plots the internal power distribution for the Top500 starting with the first release. On log-log axis, the straight lines appear to have a Zipfian distribution [55]. The plot is based on [4], where the author pleads for such a distribution.

The slope of the lines is almost imperceptibly shallower than -1, suggesting a very slight tendency towards diversity rather than redundancy; but overall an excellent balance. The fact that the slope remains relatively constant throughout time shows that the balance between diversity and redundancy is kept at optimal levels, with very few 'events' for the head of the list.

However in Feitelson [5] it has been argued that the model is deteriorating. The top of the list is a little bit too slow, and the bottom ranks grow up in performance faster than they should. According the Zipf law, the product between rank and frequency should be constant. Fig. 2.11 – Feitelson [5] – shows the Rmax axis multiplied with the rank. Any data set in agreement

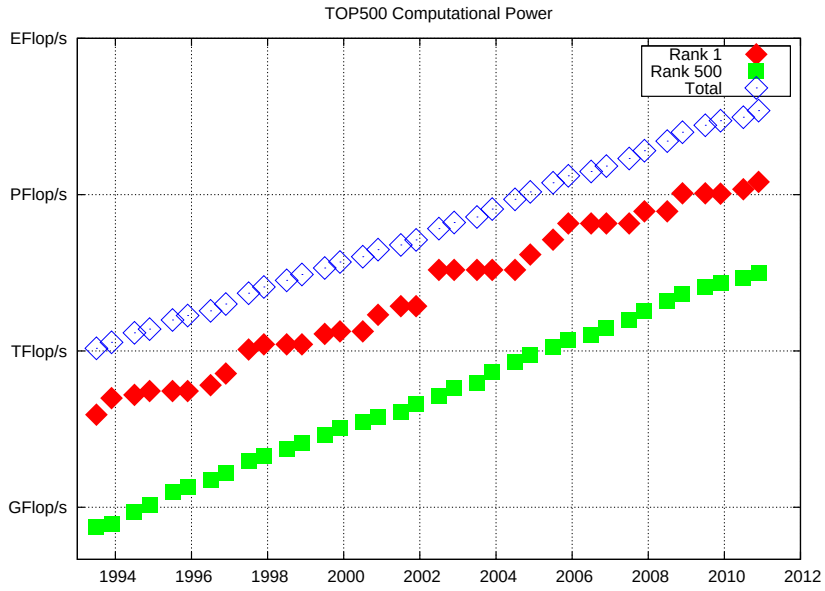


Figure 2.9: Top500 Performance Growth – based on Strohmaier and Meuer [3]

with the Zipf law should produce straight lines with a slope of 0, on semi-log axis.

Starting with 1998 the slopes are badly diverging from what they should be. The model is slowly deteriorating, suggesting that improvements are necessary. The smaller ranks show that they are more biased against the power law; a Zipf-Mandelbrot law could be more appropriate.

Statistical studies are necessary in order to understand the future development of the supercomputing industry. Even if the Top500 is not exhaustive nor complete, it clearly stands-out for the world-wide high performance computing power. Understanding the correct phenomena that takes place within the supercomputer industry, is a key to projecting its future impact on the scientific community.

Strohmaier and Meuer, for instance, introduced 4 classes of computer architectures in order to apply Bell's law of computer class formation to the Top500; they used the taxonomy in Table 2.1 [9]. The law described in 1972 states that every decade, technological advances enable a new, usually lower priced computing platform to form. Once formed, each class is maintained as a quite independent industry structure [56].

Bell also identified three directions of development for computers [56]:

- Constant price and increasing performance of an established class

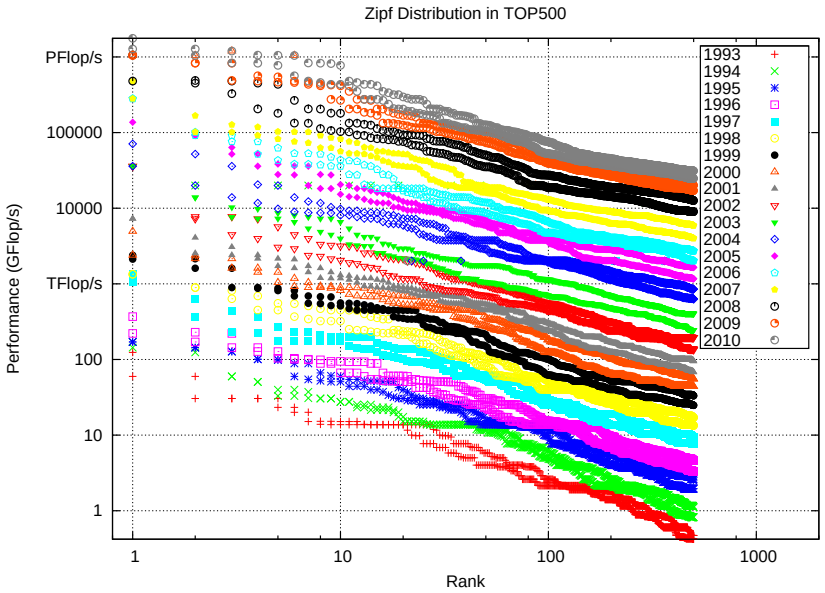


Figure 2.10: Zipfian Power Law of Top500 Supercomputers – based on Ripeanu [4]

Data parallel systems	Vector Cray Y-MP,X1, NEC SX; SIMD CM-2	1970-1990
Custom scalar systems	MPP T3E,XT3,IBM SP Scalar SMP/Constel.	1980-2000
Commodity clusters	PC cluster, Blades	1990-2010
Power-efficient systems	BG/L or BG/P low-power systems	2000-2020

Table 2.1: Bell's Law applied to Supercomputing – Meuer [9]

- Supercomputers: a race to build the 'largest' computer of the day
- Novel, lower priced 'minimal' computers

The Green500 list began in 2006 as an extension to the Top500, and redefined system performance according to the amount of FLOPS per Watt [57]. An earlier trend, of 'supercomputing in small spaces', sprang another branch of evolution [58]. It is expectable to see that BlueGene-like technology incorporates both of these trends, and that massively parallel processor (MPP) systems of this kind are going to emerge and dominate the market that is now ruled by beowulf clusters.

With the MPP market becoming more and more accessible, the technology

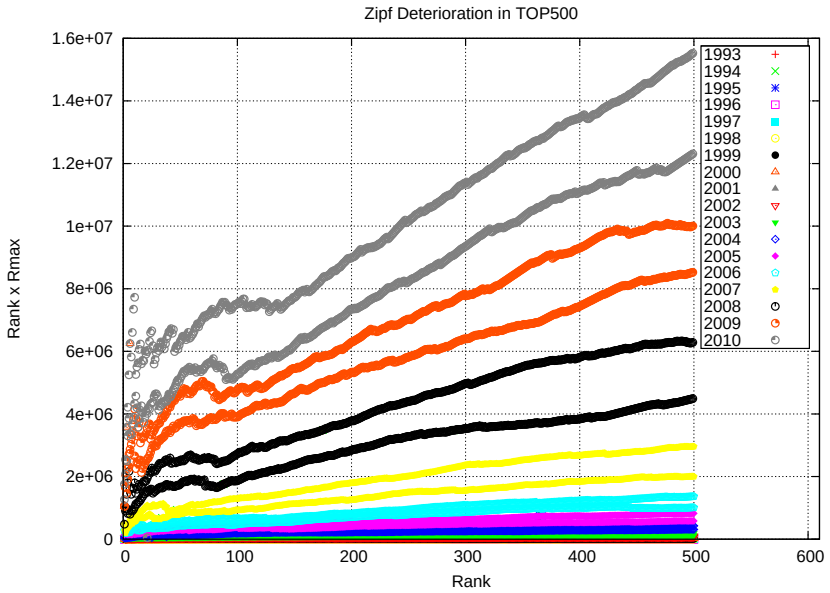


Figure 2.11: Distribution Inside the List – based on Feitelson [5]

shift in computer architecture should become steeper. The data from the Top500 list appears to be in agreement.

Fig. 2.12 displays the evolution of the main supercomputer architectures as reported to the Top500 database.

The symmetric multi-processor (SMP) rolled out before the millennium event, leaving space for clusters and constellations. Clusters are, for now, the dominant architecture with more than 80% shares. However, in 2010 they started to give up to MPP, which began to display a subtle growth in popularity. This is in fact in agreement with Table 2.1, which states that a new class of power-efficient systems must be emerging.

Microprocessors reign over the Top500. Fig. 2.13 shows the decline of the vector architecture. In agreement with Amdahl's law, microprocessors have become much easier to produce and replace. There is an upper limit in the maximum number of processing elements (PEs) that can be incorporated in a system, and that may give chances for modest recurrences of the vector machines and cluster-like constellations – an in-depth study is presented in Feitelson [54].

High performance computing is not only mature enough, but indispensable for supplying the modern scientists and engineers with tools, that open doors which have never been opened before. The closure condenses the principal lines of sight from this chapter, and concludes the subject.

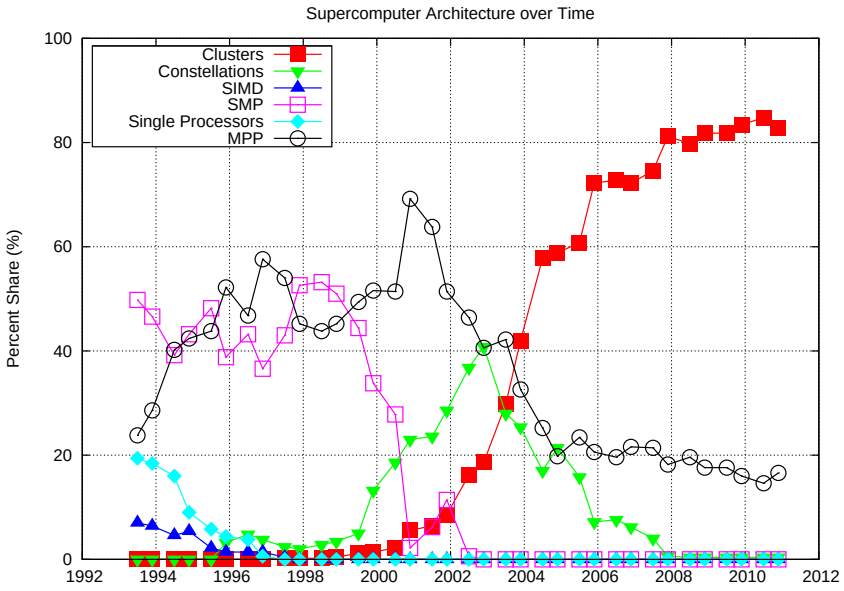


Figure 2.12: Supercomputer Architectures in TOP500

2.3 Closure

In the closure, the background in High Performance Computing is synthesized with computer science in mind, through the new, emerging field of computing in science and engineering.

High Performance Computing is a young, evolving, interdisciplinary domain, but is here to stay. It involves many other self-sustaining fields of research, intertwined at the confluence between computer science and applied mathematics. All of the scientific and engineering fields have started to rely on supercomputing for venturing into new frontiers: fluid dynamics, magnetohydrodynamics, electromagnetism, particle physics, chemistry, astrophysics, electrostatics, mechanics, with no less importance for HPC applications in art, architecture, and the social and life sciences.

From a computational standpoint, computational science and engineering (also known as CiSE) is the name of the game. The new field of CSE, for the first time, is bringing computer scientists together with researchers of different expertise, in order to cope with the grand challenges that arise when new frontiers have to be explored.

Armed with a correct understanding of the field, the following chapter investigates the available state of the art methods for dealing with large numerical simulation data.

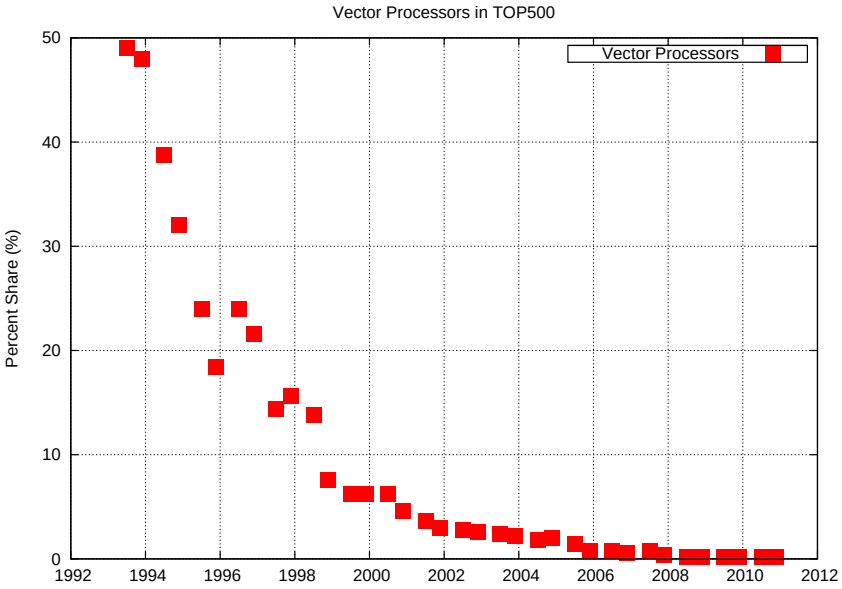


Figure 2.13: Decline of the Vector Processor in TOP500

3

State of the art in dealing with large numerical simulation data

3.1 Methods from computer science

There are many attempts to deal with the problem of large numerical simulation data. This section is an extensive study which outlines the most important approaches originating from the computer science field.

The U.K. Research Councils define e-science as 'large-scale science carried out through distributed global collaborations enabled by networks, requiring access to very large data collections, very large-scale computing resources, and high-performance visualization'. One may be tempted to further elaborate; science journalist George Johnson concludes his apologia: 'as research on so many fronts is becoming increasingly dependent on computation, all science, it seems, is becoming computer science' [59].

Europe is well aware of the importance of having a well-structured HPC infrastructure, and the urgent need to exchange experiences and know-how across the Union [60]. On the other hand, deep scientific computing requires large, deep data [61]. Therefore, the scientific world is entering a new paradigm, that of data-intensive science [62]. As noted by Gray et al. [62], analysing this data requires methods that can deal with huge datasets, and can find very subtle effects overlooked in the previous measurements. The authors believe that most science happens when data is examined in new ways.

In Gerndt et al. [63], the authors develop a virtual reality (VR) computer aided surgery (CAS) system that resolves the airflow inside the patient's nasal cavities, in order to support the medical stuff during rhinosurgery. The anatomy is extracted from computer tomography (CT). Besides being time-consuming, the simulation faces serious problems to deal with the data in real time. The authors are investigating cache and prefetch operations for more advanced simulation runs. Similar CAS examples show how critical it is, for

numerical simulation data, to become available within real time constraints, when the context demands it.

Gray et al. [62] argue of why scientists avoid databases. The author of this thesis believes that the explanations are subjective. The truth is that database software, in general, lacks the functionality and features that can make it attractive, and fast enough for science. The critique is extended by Burns et al. [64], who declare scientific databases to be an 'orphan in the database community'.

One important feature that is not yet fully developed, for example, is the ability to handle time-varying data at high resolutions.

Based on experience and the developments in literature, the author of this thesis believes that an ideal database system for CSE has to meet these very basic constraints:

1. Provide high performance, distributed, parallel query processing;
2. Have an object-relational oriented structure;
3. Allow for very large number of levels for compressed metadata;
4. Provide large, multi-indexing capabilities;
5. Default to intelligent, compressed multi-scale, multi-resolution storage;
6. Provide smart, time-travelling, multi-resolution query processing;

Many of these features are already implemented in unconnected projects. Both HDF [65] and netCDF [66] provide many of these requirements and are used by the scientific community; they are the most popular (and portable) file formats in CSE, and already present the abilities of simple database systems. The parallel version of netCDF provides high performance implementations [67]; many other projects follow the same blueprints.

We may one day realise that such a database system could evolve into something that is closer to artificial intelligence (AI) and cyber-scientist assistants, for all the complexity and processing power involved. In any case, time-travelling query processing at arbitrary resolution levels is key for developing mature, dedicated databases in CSE. Until then, proprietary and custom file formats are going to prevail, maintaining the diversity of unconnected neurons that share the same bazaar.

Databases are made of files, and files are organised into filesystems. The literature bursts with multi-indexing, multi-scale metadata approaches for handling scientific data storage, which sit on top of database systems. These in turn can benefit from parallel filesystems [68].

Korsemeyer and Thompson [69] see the Web and the Grid, as part of the Internet infrastructure, the de facto medium for sharing raw scientific data. However, the dedicated middleware, such as a specialised database system, is acknowledged to be missing.

The natural approach to access high performance resources is centralistic. This poses several problems. Strategic locations for data warehouses that can be counted on fingers are easy to spot, and vulnerable targets. In Europe, they are supposed to be 5, with several backup regions. Destroying the information from only one of such places may prove to be more horrific on the long-term than the direct bombing of hospitals. Many other scenarios can render the data inaccessible, and suspend the research activities on large scale levels.

It is essential that smart techniques are developed in order to:

1. Retain easily portable, minimal information from the huge datasets;
2. Allow for quick reconstruction of the key original data features;
3. Cheaply empower independent researches to continue the scientific investigations;

Hiraki et al. [18] propose a 'data reservoir' approach, which uses multi-gigabit backbone networks to connect research facilities together. The implementation is based on the iSCSI protocol [70].

The people from numerical weather prediction (NWP) face serious challenges when dealing with simulation data. The solution usually involves some sort of Grid transport [71]. On the other hand, when the forecast has expired and can be adjusted with experimental measurements, the numerical simulation becomes obsolete.

The Global Telecommunication System (GTS) is a point-to-point and point-to-multipoint network that interconnects meteorological telecommunication centres. The most popular data format is the GRIdded Binary (GRIB), with various versions and improvements [72].

Skidmore et al. [73] propose a virtual notebook environment, implemented on top of web services, aimed for providing scientists with support for collaboration, and easy management of computational experiments. It is an electronic version of the paper-based lab notebook. Experiments can be launched using a visual specification language. The prototype is tested with neuropsychological data.

An intelligent data service system has been developed at the Sandia National Laboratories for internal use [74]. The key principle identified in the paper is that very large data sets should be processed in situ, and not moved

around. Based on this idea, smaller data objects representing key features are extracted and tracked inside the facility, in order to provide scientists with an easy way to transport, and interpret, information. After all, the final integrator of visual representations is the human brain.

Monti et al. [75] use peer-2-peer technologies to offload simulation data from HPC centres. By using an overlay of mediator nodes, they are able to reduce the transfer time by almost an order of magnitude. Furthermore, Vazhkudai et al. [6] – see Fig.3.1 – introduce the FreeLoader, a distributed local area network (LAN) storage system, to cope with data deluge from experiments, simulations, and apparatus.

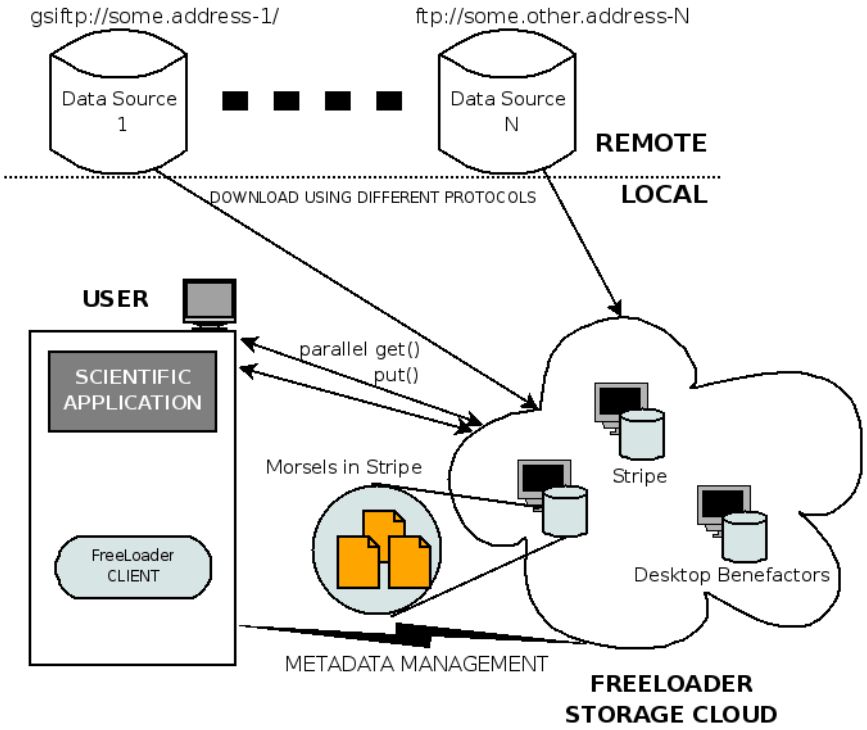


Figure 3.1: FreeLoader Storage for Scientific Data – based on Vazhkudai et al. [6]

The final stage in the data analysis workflow usually requires intensive post-processing and visualisation operations at local user computers, most likely local desktop workstations and personal notebooks. End-user hardware has an increasing computational performance, but is ill-equipped when it comes to I/O bandwidth rates. On the contrary, a large portion of the local hard disk storage is free. Based on these observations, the FreeLoader project exploits data locality by intelligent caching and prefetching of data inside the

LAN. The storage is donated by the local users – called benefactors, just like in regular peer-2-peer distributed systems.

End-user hard drives have lower I/O rates than common local area networks. This enables the clients to use parallel get() and put() operations from different benefactor machines. The data is organised into stripes (chunks) and fragmented into smaller morsels.

The FreeLoader borrows desktop storage scavenging, from peer-2-peer systems, parallel I/O and data stripping from parallel filesystems, and caching from cooperative caching systems Vazhkudai et al. [6]. Hot data always replaces seldom accessed information, and when the requested bits are not available inside the LAN, they get downloaded from remote locations, repositories, and archival sites. This makes the FreeLoader approach well suited for implementation at institute and facility level. One can easily envision grids of FreeLoader systems connected together, when additional security features are in place.

Huge data sets are only useful if they can be visualised. Visualisation maps large quantities of data into graphical representations, such that they exploit the superior visual processing capability of the human brain, which translates the visible spectrum into experiences that can be analysed inside the internal, associative neural network. That is, visualisation enables the human brain to quickly draw inferences and detect patterns. Developing powerful visualisation tools plays a key role in CSE [76].

Unat et al. [77] present adaptive subsampling (coarsening) techniques, for the compression of scientific data. Just like wavelet compression, which is reviewed in the next section, adaptive coarsening is a multi-resolution technique. Compression ratios of at most 8:1 are achieved for turbulent flow numerical simulations. These can be adjusted to be in agreement with the necessary accuracy. The method however has not been compared with adaptive mesh refinement techniques – like those described by Jasak [78]. The main goal of adaptive coarsening is to achieve compression, and for adaptive mesh refinement the goal is to optimise the solution grid; however, in both cases, the essential features of the simulation are retained, according to specified accuracy constraints.

Techniques from high performance visual data analysis and scientific data management are combined in [79], and applied for accelerator science. In [80], the authors present multi-resolution bitmap indexes as a method to improve the performance of scientific databases. Data from earth and rocket science is used, with parallel implementations.

Remote visualisation of a numerically simulated turbulent jet is described in [81]. The authors propose the usage of 'visually lossless' (lossy) video compression to transport images over wide area networks in real time. The

data is rendered using a local cluster, but partial rendering is suggested for user clients with graphic capabilities.

Computer engineering has been developing smart look-ahead techniques for a long time, in order to alleviate performance bottlenecks. Memory caches are good examples for exploiting data locality [82]. Texture compression has also been used in computer graphics for a while. With embedded systems becoming more and more ubiquitous, several code-size reduction techniques have started to take shape [83]; finally, bus level compression boosts the data rate for intensive applications [84].

Out of this primordial soup comes the idea of floating point compression, using predictors, context dictionaries, value differencing – for time locality – and, when geometric information is available, spatial locality.

Floating points, especially double precision, are widespread in scientific use. Bassiouni [85] explores the challenges and benefits that arise when compressing scientific and statistical databasess (SSDBs). Differencing and run length encoding (RLE) are the most frequent techniques.

Ghido [86] introduces an algorithm for lossless compression of IEEE floating point audio. He claims that the method, based on integer linearisation, is also applicable on different types of data sets; however, the algorithm is designed for 32-bit audio and needs to be extended for double precision data.

The work of Xie and Qin [87] is also single precision, but is aimed for compressing seismic data. The method uses a differential predictor with adaptive contexts. Engelson et al. [88] on the other hand, use a predictive coder which is based on high-order polynomial extrapolation, and they tests it with ordinary differential equation (ODE) data.

When geometric data, like mesh information, is available, spatial coordinates can also be compressed and smarter locality algorithms be devised. Such is the work of Lindstrom and Isenburg [89], where the authors use a Lorenz predictor to produce high throughput compression rates. Without the mesh information, the compression ratio of other algorithms may depend on how the structure is transited by the encoder.

Starting with 2006, floating point compression for numerical simulation data has started to mature. Ratanaworabhan et al. [90] use a differential finite context method. The original method was introduced in hardware design to increase the Instruction Level Parallelism (ILP) of microprocessors [91]. The compressor works for raw streams of double precision floating point (FP) numbers, without requiring any geometric data. Later on, in [92], they introduce FPC, a compressor for linear streams of 64-bit floating point data. The method is finally polished and disseminated via [7]. FPC achieves compression ratios varying between 98% and 6.64%, the later being obtained for numerical plasma simulation data. Also, in terms of speed, it outperforms

state of the art competitors on the given data set, and its simplicity allows it to be implemented in very high throughput hardware devices.

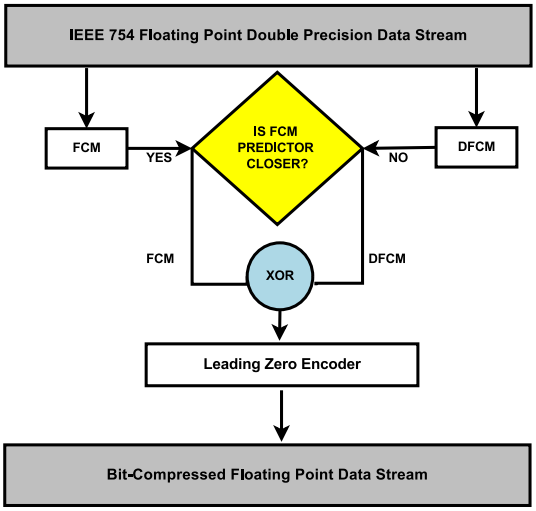


Figure 3.2: FPC: The Floating Point Compression Algorithm – based on Burtscher and Ratanaworabhan [7]

Fig.3.2 is based on Burtscher and Ratanaworabhan [7], and describes the FPC algorithm. The authors use both fcm [93] and dfcm [91] for predicting data inside the floating point stream. The predicted value that is closer to the original is going to be XOR-ed against it, so that as many bits as possible are set to zero. The number of leading zero bits is encoded as a 3-bit value. An additional bit is added to distinguish between the predictors that were used. The resulting nibble together with the 'residual' non-zero values are sent to the compressed bit stream in data blocks, using interleaved positions to obtain byte-level efficiency.

The FPC algorithm is very easy to implement on both hardware and software, is fast, and can be used on any sort of double precision floating point data, without prior knowledge of the content type or any metadata information. This makes it suitable for implementation in numerical simulation storage formats and makes it a preferred choice for combining it with different technologies (including the FreeLoader).

On the other hand, turbulent data is harder to predict, in space, and the common octree data structures [94], that are used to hold mesh field information, are stored using geometric numbering schemes – therefore, compression ratios for unprepared streams of spatial vector and scalar fields, holding turbulence data, are going to be suboptimal.

The parallel version of FPC, pFPC, is designed for speed, and is able to trade off compression ratio for throughput [95]. Compression is usually more efficient when the number of parallel threads reflect the data dimensionality. For vector fields, data dimensionality is 3 – best compression factors are obtained when data is interleaved on 3 channels. It is critical to pre-process the data before compression and transform everything into a properly shifted floating point stream. How to obtain the optimal permutation, especially when the simulated phenomena are unsteady, is a matter yet to be investigated, and it is not the scope of this thesis.

The self-tuning FPC, gFPC, uses genetic algorithms to adapt its hash function, and it is four times slower than the original, but achieves slightly better compression rates [16].

Sano et al. [96] use field-programmable gate array (FPGA) hardware to implement another compressor, this time using a 1D cubic predictor which is more efficient than the 2D Lorenz method. Tomari et al. [97] use software for the compression, and application-specific integrated circuits (ASICs) for decompression. The mantissa is not compressed. Exponents of 11 bits, are packed together with the sign in 12-bit words, which, in turn, are grouped into 4-word vectors. Considering 64 vectors at a time, data is transmitted in three possible scenarios related to the context dictionary: near match, perfect match and uncompressed. When the block of vectors looks similar with previously seen data, they can be compressed by an index and a small difference if necessary. Mantissa remains untouched, and can be encoded with usual encoding schemes.

As a continuation of pFPC, GFC is designed by O'Neil and Burtscher [98] for massively parallel Graphics Processing Unit (GPU) computing. Since GPU devices are usually integer oriented, the authors interpret the floating point data as a stream of integers, in order to gain speed.

The school of computer science has been producing valuable solutions based on floating point compression and peer to peer systems. The remaining of this chapter presents related works from applied mathematics, and draws the final remarks on how the new concept which the author proposes connects with the state of the art in different scientific domains.

3.2 Methods from applied mathematics

The previous section has explored the methods for dealing with large numerical simulation data from the computer science perspective. However, since computational science and engineering is a borderline field of study with strong mathematical foundations, it sounds reasonable to also review

the available approaches originating in applied mathematics.

The discovery of discrete wavelet transforms (DWTs) have revolutionised fields like digital signal processing (DSP), and image compression. The FBI uses Daubechies wavelets to compress grayscale fingerprint data up to 5%–8%. [99]. But wavelets have also started to be used in CFD, for modelling turbulence [100].

Kim et al. [101] propose a method for dealing with ultra large scale direct numerical simulations (DNSs); the procedure also facilitates fast domain decomposition and easy implementation of compression.

Lossy, multiresolution CFD compression, using wavelets, has slowly started to spread. Kang et al. [102] propose a hybrid method using super-compact wavelets. They achieve compression ratios between 14.5:1–34.8:1 with acceptable losses in accuracy. They also propose that such methods should be used for remote visualisation of large scale numerical data.

In order to display the idea behind wavelet compression, the Haar wavelets are used, based on the explanation from [103] and the work in [104]. Consider the continuous function in (3.1).

$$f : [0, 1] \mapsto \mathbb{R} \tag{3.1}$$

The goal is to approximate f with piecewise constant functions. The natural way would be to split $[0, 1]$ into a number of intervals, and to approximate $f(x)$ on each of these intervals with the average value of f on that interval. For simplicity, consider the interval $[0, 1]$ to be split into $N = 2^k$ intervals of equal length.

These intervals are all of length 2^{-k}, so the average value of function f on the n^{th} interval is given by a_{f_n} in (3.2); k is called the discretisation scale.

$$a_{f_n} = 2^k \int_{n2^{-k}}^{(n+1)2^{-k}} f(x)dx \tag{3.2}$$

Now that the data has been discretised into a_{f_n} values, let A_n be described by (3.3) as a set of discrete entries that require compression.

$$A_n = \{a_{f_n} \mid n = 0, 1, ..2^k - 1\} \tag{3.3}$$

Define the directed distances d_j as in (3.4), with $j = \{0, 1, .., 2^{k-1}\}$,

$$d_j = \frac{a_{f_{2j}} - a_{f_{2j-1}}}{2} \tag{3.4}$$

and the sum s_j as in (3.5).

$$s_j = \frac{a_{f_{2j}} + a_{f_{2j-1}}}{2} \tag{3.5}$$

The process is reversible since $a_{f_{2j}} = s_j + d_j$, and $a_{f_{2j-1}} = s_j - d_j$. This is just a one level transformation. The full wavelet transform must be performed in a recursive manner, such that the s_j values from step m must become the A_n values of step $m + 1$.

If $s_j^{(l)}$ and $d_j^{(l)}$ denote the appropriate values at level l, then $s_j^{(0)} = A_n$, and the values for the higher levels are recursively given by (3.6) and (3.7).

$$s_j^{(l)} = \frac{s_{2j}^{(l-1)} + s_{2j-1}^{(l-1)}}{2} \tag{3.6}$$

$$d_j^{(l)} = \frac{s_{2j}^{(l-1)} - s_{2j-1}^{(l-1)}}{2} \tag{3.7}$$

The inverse transform is also given by (3.8) and (3.9).

$$s_{2j}^{(l)} = s_j^{(l+1)} + d_j^{(l+1)} \tag{3.8}$$

$$s_{2j-1}^{(l)} = s_j^{(l+1)} - d_j^{(l+1)} \tag{3.9}$$

Compression is achieved by only storing the differences $d_j^{(l)}$ at each level, the number of which is halved during each iteration. The topmost level is $L = log_2 N = k$, having a single distance coefficient $d_0^{(L)}$ and a summation coefficient $s_0^{(L)}$ that also has to be stored. L is called the level of recursion and may be different from k. The discretisation scale is only important at a different stage, and is not necessary when the data is already in a numerical form.

It is easy to notice that wavelets allow for multiscale levels of detail to be decomposed from a discretised signal. That means that if not all of the levels of details are necessary, one can simply achieve lossy compression by limiting the recursion level L. The nice effect is that wavelet decomposition will tend to preserve the cumulative energy from a given dataset, and thus allow for an implicit version of feature extraction.

Another effect is that the difference data can be encoded on a lower number of bits. Different encoding schemes can be deployed, and very small coefficients can be neglected. Other wavelets are more efficient at compressing data, and it is still a matter of investigation to chose the right model for a given data set.

Trott et al. [105] for instance use wavelet compression on 32-bit FP numbers and state that only 3% of the data is necessary in order to produce a decent visualisation. While the brain integrates the visual information, the rest of the data can be downloaded in the background, to obtain the original uncompressed version.

CFD simulations often present vorticity and shocks which are very hard to capture in progressive decompositions. However, wavelet based methods have the ability to capture the time-frequency relationship with ease, and are suitable for such discontinuities in the data stream. Standard Huffman coding is used for the compression of the wavelet coefficients.

Wilson [106] claims that 'lossy compression is the only choice for turbulence data'. Later in [107] his findings conclude that data from turbulence simulation does not usefully compress with lossless methods. Considering 8 bits per pixel to be the common practice data rate for visualisation, he proposes a 4:1 compression ratio. However, not all scientific visualisations can be reduced to the necessary bit rate.

Giles [104] presents a report for wavelet compression of unsteady CFD data. He compares the advantages over Fourier compression and arguments that 'the local nature of wavelets means that a discontinuity in the data only affects the amplitudes'. He also considers CDF 3,1 wavelets for better performance than the popular Haar ones.

Amaratunga [108] applies wavelet compression to reduce the in-core kernel matrix storage for large-scale simulations of 3D integral problems.

Numerical data deluge is not only a problem during post-processing operations. Several matrix storage formats have been proposed to deal with this issue [109]. Liu [110] proposes a compact row-oriented storage scheme for Cholesky factors. Andersen et al. [111] implement a Cholesky solution of symmetric positive-definite systems of equations, using packed storage.

Sommeijer and der Houwen [112] use techniques for low-memory programming in order to develop a numerical algorithm for the solution of parabolic equations. Shampine [113] uses similar techniques for Runge-Kutta codes.

Structural analysis is the computational science that deals with the analysis and simulations of numerical stress forces within mechanical structures. Most of the phenomena can be described by linear equation models, unlike complex, industrial CFD, and they have immediate, practical applications in engineering.

The roots go back in the 1960s, in aero-space engineering, when it was easier to analyse the stress forces that appear in aircraft structures during different flight regimes, rather than compute very basic streamlines around the fuselage.

At that time, computer memory was very limited, and scientists were forced to develop ingenious workarounds to get the job done. One of the common practice was static system condensation, also known as substructuring [14].

Static system reduction (condensation) aims to split a system of equations like in (3.10) into subparts that can be dealt with independently.

$$Ax = b \tag{3.10}$$

Substructuring methods are based on a more generic mathematical concept, called 'partial matrix inversion' or 'partial elimination'. The goal of substructuring is to condense the system of equations to lower numbers of unknowns. This can be done in many ways, but for the sake of simplicity the explanations from [114] are used as follows.

In (3.10) the matrix of the system is $A \in \mathbb{R}^{n \times n}$, and $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$ are vectors with x holding the unknown variables.

The system can be split into blocks like in (3.11).

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{13} & A_{14} \end{pmatrix} \begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix} = \begin{pmatrix} b^{(1)} \\ b^{(2)} \end{pmatrix} \tag{3.11}$$

If A_{11} is regular, then $x^{(1)}$ can be expressed in terms of (3.12).

$$x^{(1)} = A_{11}^{-1}(b^{(1)} - A_{12}b^{(2)}) \tag{3.12}$$

Now substitute (3.12) in (3.11) to get (3.13), which only contains half of the unknown vector.

$$(A_{22} - A_{21}A_{11}^{-1}A12)x^{(2)} = b^{(2)} - A_{21}A_{11}^{-1}b^{(1)} \tag{3.13}$$

Substructuring and superelements are also useful for commercial or strategic reasons, when different engineering teams have to complete the analysis of different natural subparts independently.

The whole idea behind iterative numerical methods is to avoid inverting the matrix in (3.10), and produce an approximate solution through repeated matrix-vector multiplications. This makes substructuring unfeasible for large scale CFD simulations since it is based on the inversion of matrix subparts. Smarter techniques, like partial Gauss elimination [41] or complex approaches for large sparse system matrices are known. However, the complexity of CFD phenomena requires a solution for very large algebraic systems, and that makes similar attempts unfeasible. This is, in the author's opinion, the main reason why partial elimination techniques have not been developed in CFD. The nonlinearity of the turbulent flows translates into many degrees of freedom which need to be taken into account. Also, substructuring in the middle of a fluid flow, in order to capture interesting features, makes the boundaries completely dependant on the global simulation. The procedure, besides being extremely expensive, would have to be repeated during each time step. In structural engineering, the subparts are carefully chosen in order

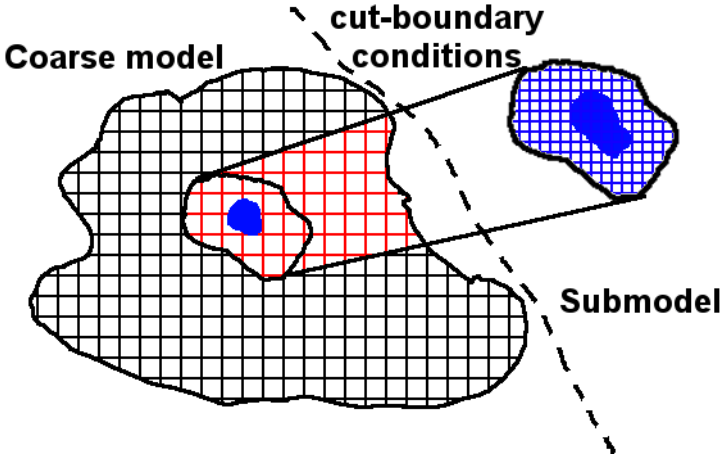


Figure 3.3: Submodelling

to minimise the boundary contacts, and that's when following the natural components becomes standard practice: wings, tails, and cockpits.

Another method, sometimes called 'global-local' analysis, is submodelling [15]. Global-local analysis is performed in stages – see Fig.3.3. First, a coarse simulation of the global model is performed. Out of it, boundaries are cut and a submodel is extracted with a higher resolution level. In order to transfer the data from the coarser mesh to the finer mesh, interpolation is used. The submodel is then analysed independently from the global model. As a side effect, data for the cut-boundary conditions is orders of magnitude smaller than the global simulation fields.

However the procedure is very restrictive. Submodelling can only be applied when the local phenomena is uncoupled with the global simulation. In structural analysis, for example, this is known as the Saint Venant's principle, and it also works for electrostatics. The principle states that 'the strains that can be produced in a body by the application, to a small part of its surface, of a system of forces statically equivalent to zero force and zero couple, are of negligible magnitude at distances which are large compared with the linear dimensions of the part' [115]. In more colloquial words, if a sheet of paper with a circular hole in the middle, is exposed to marginal constant forces which tend to elongate its length, the stress induced far away from the hole is negligible, as opposed to the stress concentrated around the hole which will produce cracks in the material. Impurities can take the role of holes or cracks, so this principle plays a key role in material stress analysis.

But what is more important now in relation with storage requirements, is that submodelling can not be applied when this principle does not hold.

Not only does this principle not hold in CFD, but submodelling nonlinear equations in the middle of a turbulent flow has never been attempted. One of the methods presented in this thesis is based on submodelling. Mass and flux conservation have to be taken into account, and one has to develop methods for reconnecting the equations together again after the submodel is extracted. During unsteady simulations, errors are introduced at each time iteration, so the procedures have to be repeated. The complexity of the proposed method goes way beyond classic submodelling techniques.

This section produced a synthesis of what applied mathematics has to offer for dealing with large numerical simulation data. The conclusion will recapitulate the limitations in the state of the art and will state the problem that is being challenged by the thesis.

3.3 Conclusion

This chapter has introduced a thorough survey of the state of the art methods for dealing with large, numerical simulation data. The final section closes the chapter by recapitulating the limitations from the current solutions, and by stating the problem that needs to be solved in the next chapters.

Large sizes of data have always posed serious problems to the computer science community. This still holds true when large sizes of data used to mean a few kilobytes. The ability to distil information may be of more importance than the ability of simply producing or acquiring it. Truth cannot be derived from facts, it proceeds from meaning. Just like Tycho Brahe used to develop false planetary models based on the very thorough observations that he made, it took Kepler another decade of hard work, using the very same information, to discover the laws that later enabled Newton, to comprehend the laws of universal gravitation.

Several techniques have been developed that are clearly driven by the unmediated desire to comprehend the large datasets at hand, either produced by simulations, or acquired by sensors. All these approaches relay on the fact that the human brain is the final and ultimate computer, capable of integrating and processing everything. And for now the best way to interface the brain with the data is through visualisation, by bombarding it with controlled fluxes of photons.

Duque and Legensky [116] use in-situ post-processing and visualisation for large-scale unsteady CFD data. Driven by the desire to understand the results, they consider that often enough it is only necessary to extract isosurfaces and transport them to the client side for further post-processing. Also, they make use of intelligent GPU texture mapping so that unnecessary bus traffic

is avoided.

Yu et al. [117] propose in-situ visualisation as a scalable way for dealing with very large, highly intermittent and unsteady, turbulent combustion phenomena. Ignition and extinction are given as examples.

Kwan-Liu et al. [19] propose smart in-situ visualisation techniques as the solution for ultra large numerical simulations. They also deploy feature extraction techniques to reduce the data to the minimal information that the brain needs to process in order to integrate the new knowledge. A survey of feature extraction techniques and challenges is presented by Frits et al. [20].

No matter how smart, feature extraction techniques are still in early stages of development and unless they are tightly coupled with intelligent SSDBs, as defined in the previous sections, they will fail to reach full potential.

For instance, the author believes it is critical to allow the CSE user to experience with a little bit more than very basic post-processing at client side, and also to permit further investigations and refinements; AI may or may not be ready soon enough to provide such complex features. Also, no constraints should be imposed on client connectivity, and the key features – the raw intelligence – of the results, should be portable and easily reproducible with commodity hardware.

Data deluge in large, parallel numerical simulations is still one of the main problems of modern High Performance Computing. Different approaches, from different schools of thought have been attempting to alleviate the data-related bottlenecks in supercomputing, but without any decisive success. The main data bottlenecks in supercomputing can be summarised in three levels:

1. Client level, between the client and a trusted checkpoint
2. Gateway level, between the trusted checkpoint and the supercomputing gateway
3. HPC facility level, between the supercomputer gateway and the internal nodes

The current state of the art approaches only scratch the surface of the problem, without being able to dig deep into the roots of the issue, and eventually provide well targeted solutions. For this reason, the next chapter introduces a new concept, called ‘space-time window reconstruction’, by proposing two implementations for the problems in the state of the art.

‘What do you want to store these days?’

Think Correctly

4

Proposed solutions to the state of the art limitations

The methods from the state of the art in dealing with large numerical simulation results, only scratch the surface of the data deluge problem. This chapter proposes a new concept, called ‘space-time window reconstruction’, which addresses the roots of the problem.

A space-time window is an independent numerical simulation, based on a large scale version, capturing a subdomain of analysis, in both time and space. The following sections summarise the problems in the state of the art, and then introduce two possible solutions for the implementation of the space-time window concept. The main purpose is to alleviate the supercomputing bottlenecks and the data deluge problem.

4.1 Problem Statement

This section expands a synthesis on the limitations of the state of the art methods for dealing with large numerical simulation data.

The typical structure for a HPC centre is presented in [75]. Fig.4.1 depicts such an organisation with an emphasis on the main bottleneck problem that is the subject of this thesis.

During the previous chapters, in order to emphasise the limitations in the state of the art, the author compared modern HPC facilities with complex, and very expensive aquaria. One can build the most advanced U-boats inside these aquaria, and then finally realise that there is no way to move them out. The same is true for large numerical simulations that require time and expensive technology to produce.

Fig.4.1 shows that in a typical supercomputing centre some nodes are dedicated to computing, depicted as processing elements (PEs), while others are assigned to storage handling. All traffic into and out from the facility has

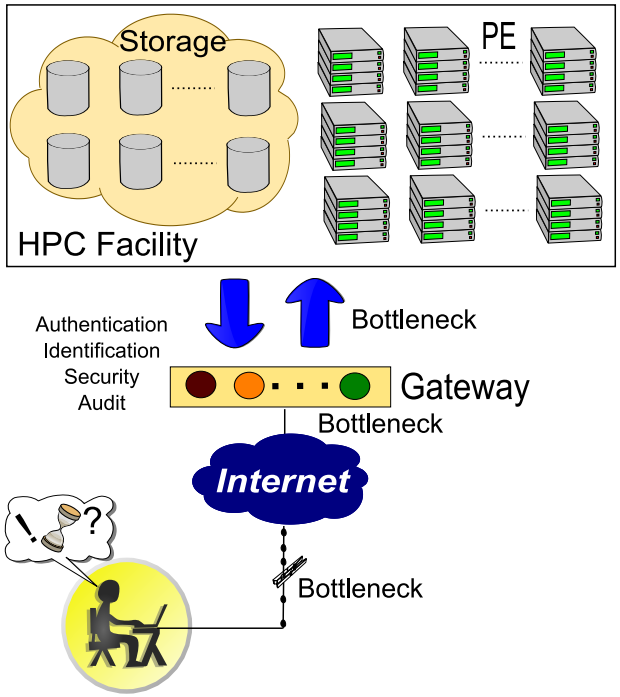


Figure 4.1: Bottleneck Problems: Typical Organisation for a HPC Facility

to pass through the gateway level. This already creates a huge bottleneck for uploading and downloading data into and from the supercomputing centre. At the gateway level, users are identified, authenticated, assigned with certain security clearances and logged for audit.

The Internet connection can vary from very good, with limited bandwidth, to very poor, with data losses and packet duplications, and show fluctuations in response times. This fact usually poses the most serious constraint on the user who is trying to remotely access the data within the supercomputing centre. The gateway's Internet connection is usually very good, but has limited bandwidth and hardware resources – so the number of simultaneous clients is limited. The third bottleneck happens when the gateway tries to access the data from the storage zones or from the computing nodes.

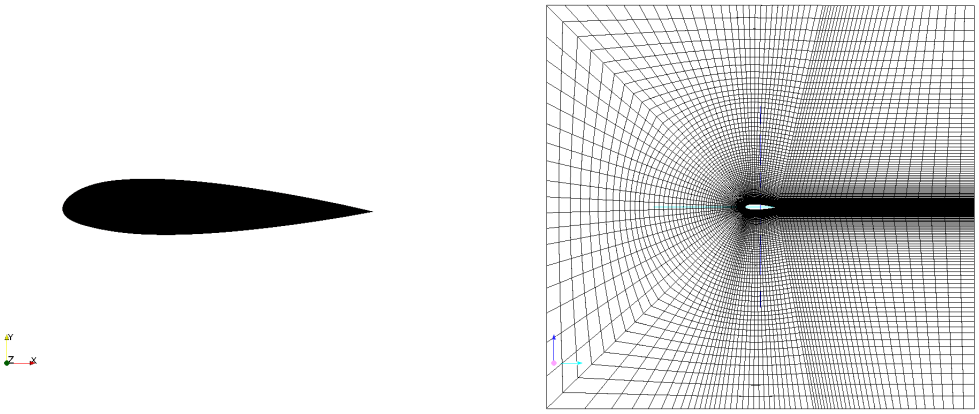
All that being said, three levels of data transportation bottlenecks are crippling the user's ability to handle the output from large numerical simulations. The thesis addresses the bottlenecks by trying to reduce the global simulation to one or more windows with minimal data, in simulation space and time.

Two solutions for implementing the space-time window reconstruction concept are proposed in the following sections; the interprocessor traffic origi-

nating from parallel matrix operations, can either be approximated through geometric interpolation (Solution A), or it can be completely intercepted and archived as in (Solution B). Parallel matrix operations are the foundation stone for large numerical applications, and before continuing, the basics must be revised.

4.2 Parallel Matrix Operations

Before continuing with the proposed methods, some background information about parallel numerics needs to be established. The general aspects of scientific computing are well described by Tveito and Winther [35], with an approach focused on computational audiences. This parallel matrix-vector refreshment is based on the thorough explanations from Quinn [8].



(a) Original airfoil (b) Mesh around the airfoil

Figure 4.2: Discretisation of the Analysis Domain

Given the domain of analysis in Fig.4.2, the original airfoil (a) is described with computer aided design (CAD) methods. In order to study the flow around the airfoil the domain is discretised by creating the mesh in (b). The equations governing the flow are going to be solved at points located at the vertices of the mesh or, for the finite volume method, at the centre of each elemental shape forming the mesh (in this case, parallelograms).

The connections from the graph in Fig.4.2 (b) are used to assemble the algebraic system of equations in the form of (4.1), where A is the system matrix and b is a vector configured with the help of known boundary conditions. These procedures are well described by Resiga et al. [109], and with computational audiences in mind by Nikishkov [36].

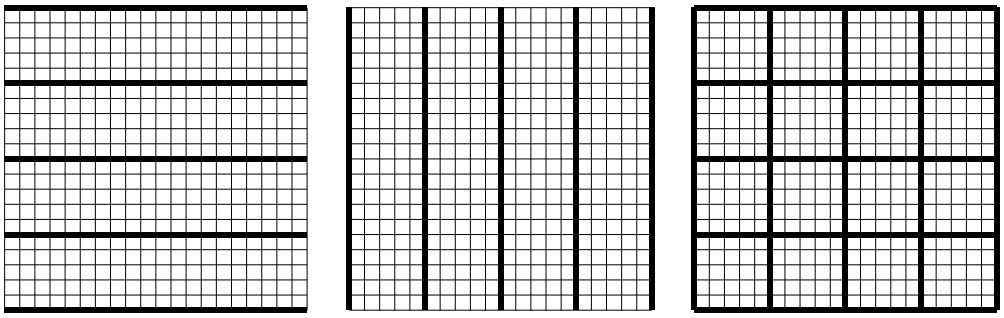
$$Ax = b \tag{4.1}$$

The system in (4.1) is solved with iterative, numerical algorithms, like the conjugate gradient [41]. One of the frequently encountered operations in such algorithms is the multiplication between a matrix and a vector, as shown in Fig.4.3, where $\forall i \in [0, m-1], c_i = \sum_{j=0}^{n} a_{i,j} b_j$ [109].

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} \\ . & . & . & . \\ a_{m-1,0} & a_{m-1,1} & \dots & a_{m-1,n-1} \end{pmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ . \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & \dots & c_{m-1} \end{bmatrix}$$

Figure 4.3: Serial Matrix-Vector Multiplication

There are three natural ways to split the matrix data from Fig.4.3, and speed-up the computations by the parallel processing: row-based decomposition, column-based decomposition, and block-based decomposition – Fig.4.4.



(a) Row-based (b) Column-based (c) Block-based

Figure 4.4: Parallel Matrix Decompositions

The vectors themselves can be split and spread across the processing nodes, or more easily they can be replicated if there is enough memory for the job. To simplify things, both vectors are considered to share the same approach of distribution among the processors. The total number of possible combinations is reduced by only considering three cases: row-based matrix decomposition with replicated vectors, column-based matrix decomposition with distributed vectors, and block-based matrix decomposition with distributed vectors. The three cases are briefly presented in the following paragraphs.

4.2.1 Row-based decomposition

In Fig.4.5, the vectors b and c are replicated inside each of the parallel processing tasks. Therefore, each processor has its own copy (one task is assigned per processing unit).

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} \\ \hline . & . & . & . \\ a_{m-1,0} & a_{m-1,1} & \dots & a_{m-1,n-1} \end{pmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ . \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & \dots & c_{m-1} \end{bmatrix}$$

Figure 4.5: Row-based Matrix-Vector Multiplication

In order to complete the multiplication, each processor exchanges the partial results with the processing world. This is done via the all-gather collective interprocessor communication, like in Fig.4.6.

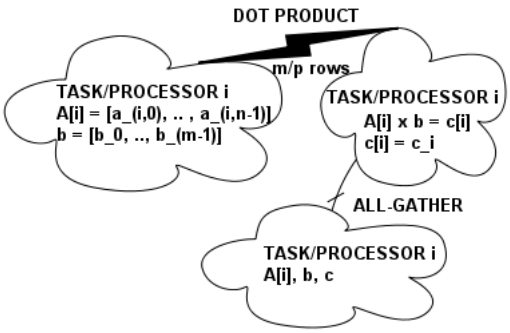


Figure 4.6: All-Gather Traffic Exchange – based on Quinn [8]

The boundary conditions (BC) that are received during the all-gather communication are given by $R_{BC_i} = (c_0, .., c_{i-1}, c_{i+1}, .., c_{m-1})$, where i is the processor number. R_{BC_i} is a set of discrete values geometrically located at various border positions in the subdomain that is designated to be resolved by processor i. The values correspond to different scalar and vector fields that are taken into account by the simulation model (i.e. pressure and velocity).

In Euclidean space, the inner product becomes a dot product. A maximum of $\frac{m}{p}$ rows are given to each processor for computation.

4.2.2 Column-based decomposition

In the column-based case from Fig.4.7, the b vector is distributed across the processors.

$$\left(\begin{array}{cc|cc} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} \\ . & . & . & . \\ a_{m-1,0} & a_{m-1,1} & \dots & a_{m-1,n-1} \end{array}\right) \times \left[\begin{array}{c} b_0 \\ b_1 \\ \hline . \\ b_{n-1} \end{array}\right] = \left[\!\left[\begin{array}{cc|cc} c_0 & c_1 & \dots & c_{m-1} \end{array}\right]\!\right]$$

Figure 4.7: Column-based Matrix-Vector Multiplication

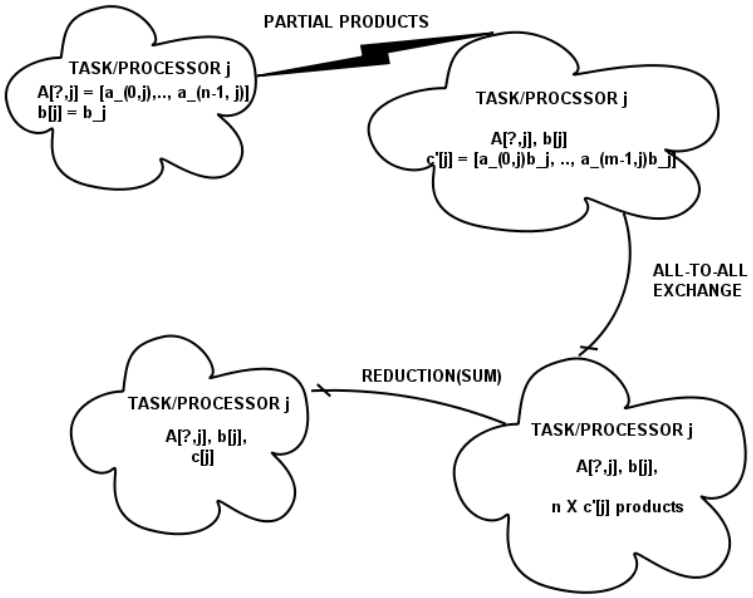


Figure 4.8: All-to-All and Reduction Interprocessor Traffic – based on Quinn [8]

Each processor computes a partial product between the column of A that it holds and its own portion of b, obtaining a column of partial products as in Fig.4.8.

The partial product columns need to be summed up in order to produce the final result of c. This is done via the reduction procedure so that the computational difficulty for addition is divided among the processing units.

The all-to-all exchange collective operation transports the partial $c^{'}$ from one processor to another, before the columns can be summed up. That is, two collective operations are used for interprocessor transfers, with the first one constructing the necessary boundary conditions (BC) for the partial multiplication, and the second one constructing the BC for local summation.

4.2.3 Block-based decomposition

In the block-based decomposition, Fig.4.9, each processor has a local submatrix with a portion of the b vector.

$$\left(\begin{array}{cc|c} a_{0,0} & a_{0,1} & \cdots \; a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots \; a_{1,n-1} \\ \hline \cdot & \cdot & \cdot \quad \cdot \\ a_{m-1,0} & a_{m-1,1} & \cdots \; a_{m-1,n-1} \end{array}\right) \times \left[\begin{array}{c} b_0 \\ b_1 \\ \hline \cdot \\ b_{n-1} \end{array}\right] = \left[\begin{array}{cc|c} c_0 & c_1 & \cdots \; c_{m-1} \end{array}\right]$$

Figure 4.9: Block-based Matrix-Vector Multiplication

After each processing element computes the local $A \times b$ product, the partial c results are summed up by reduction to produce the final vector.

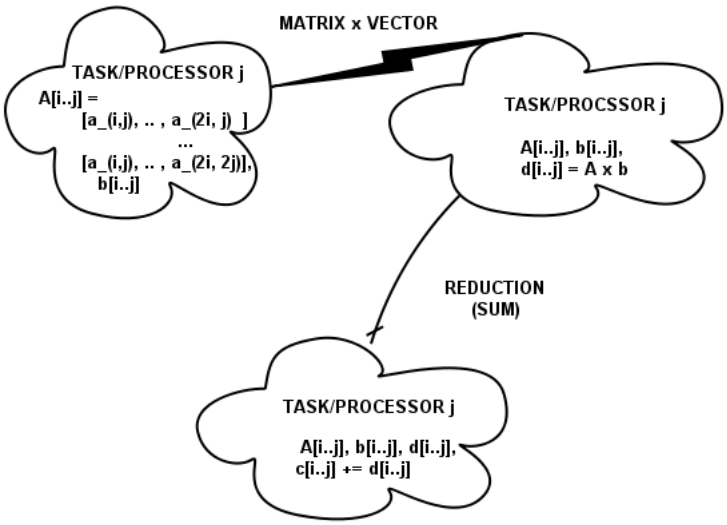


Figure 4.10: Block-based Interprocessor Traffic – based on Quinn [8]

In Fig.4.10, a scatter collective operation may be required to distribute vector b among the tasks, but is not shown.

The reduction operation can be implemented with gather calls, that require interprocessor transfers of boundary conditions (BC). This approach is more scalable than the previous decompositions. It has a reduced interprocessor traffic between the subdomains, and that makes it the preferred choice for large simulations, scaling well on an increased number of processors [118].

4.2.4 Closure

Parallel matrix operations are frequent when doing high performance

numerical simulations. The route that a message takes to reach its destination not only depends on the decomposition method, but also on the topology of the interprocessor networks, which are configured at start-up, and may actually mimic the real hardware connections.

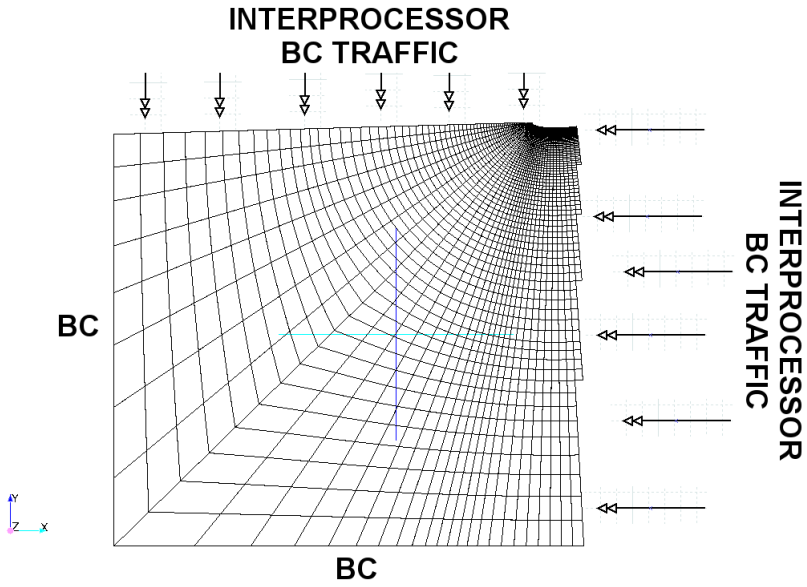


Figure 4.11: Subdomain Boundary Conditions (BC)

Fig.4.11 shows one of the 4 subdomains partitioned from the mesh in Fig.4.2(b), together with its boundary conditions at geometric locations. The border of this subdomain consists of two regions, one where the boundary conditions (BC) are given by the user from the initial problem setup, and one where the boundary conditions originate from the neighbour processors via the interprocessor traffic exchange.

If the original domain in Fig.4.2(b) is D, and the subdomain in Fig.4.11 is $S \subset D$, then the border of the subdomain S is given by $\Gamma = \partial S = BC + R_{IPC}$, where BC are the boundary conditions supplied by the user and R_{IPC} is the interprocessor traffic received from the neighbours. If the subdomain is chosen inside the analysis domain without any direct connection to the external universe ($BC = 0$), the complete boundary conditions set is supplied by the interprocessor traffic R_{IPC}.

In this thesis the author presents a new concept, called 'space-time window reconstruction'. This new concept is demonstrated using two implementations, one which uses geometric interpolation to approximate the Γ boundary conditions for a user-selected subdomain, and one which intercepts and stores

R_{IPC} in order to reconstruct the fields inside a manually defined subdomain, with the very same floating point numbers. The concept is designed to reduce the magnitude required by the simulation output, and thus to relieve the data bottlenecks. The two implementation solutions are presented in the following sections.

4.3 Solution A: Space-time window reconstruction based on submodelling

The first solution is engineered to provide maximum flexibility for the user, but instead it trades-off the accuracy levels of the reconstructed fields. The new simulation can be shifted in both time and space, so it may be necessary to revalidate the physical phenomena inside the space-time window.

Let the domain of analysis be D – Fig.4.12 – of arbitrary type. By decomposing a generic 3D domain into a set of polyhedral cells Ω_D, the discretised form, also called a mesh, is obtained. A mesh can be defined in many ways, using point lattice or graph theory concepts. For the sake of the presentation, the Ω notation is used to denote a set of geometric locations. The physical phenomena is modelled using finite fields e_j, where j denotes the index of the field in the problem space. Examples of fields are pressure and velocity, with known values corresponding to the locations in Ω space.

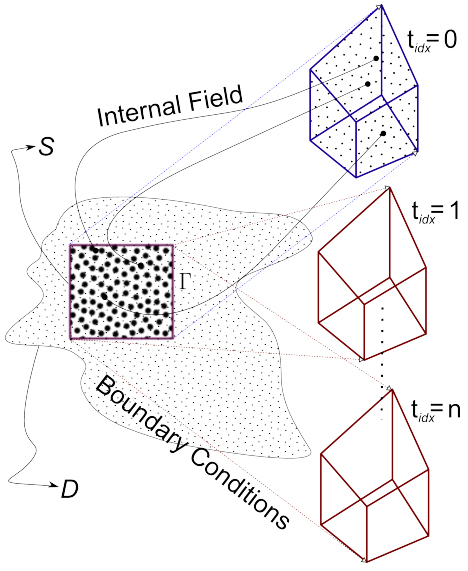


Figure 4.12: Solution A: Space-Time Window Extraction

The user defines a box-like subdomain S contained inside D ($S \subset D$),

with the boundary described by $\partial S = B$. Considering the discretised form of the equations, $\partial\Omega_S = \Gamma$. Γ is a set of discrete points defined by equation (4.2), with Γ_i being a subset i and describing discrete surfaces in 3D ($n = 6$), or line segments in 2D ($n = 4$).

$$\Gamma = \cup_{i=1}^{n} \Gamma_i \tag{4.2}$$

A new, extraction mesh Ω_{ex}, is generated to cover the subdomain S. This may or may not coincide with Ω_S. For the initial time $t_{idx} = 0$ the fields e_j in the original mesh Ω_D are transferred to the new Ω_{ex} mesh using geometric interpolation, like in Fig.4.13 and (4.3).

$$\Omega_D \mapsto_{t_{idx}=0}^{e_j} \Omega_{ex} = e_j^{'} \tag{4.3}$$

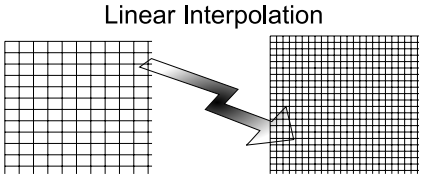


Figure 4.13: Mesh field transfer via linear interpolation

The interpolated $e_j^{'}$ boundary fields from the Ω_{ex} mesh are packed and stored into a dedicated space-time format. This procedure is repeated for each time step as shown in Fig.4.12, for $t_{idx} = 0..n$.

The complete space-time information has the property of being smaller than the global simulation, and can be easily downloaded at the user side.

Fig.4.14 shows how this space-time information is used to reconstruct the fields inside the window. A new, independent simulation is started using end-user hardware. A reconstruction mesh, Ω_{rc}, is generated inside the subdomain S which is now a standalone analysis domain, $S = D^{'}$. The $e_j^{'}$ initial fields for $t_{idx} = 0$ are transferred from Ω_{ex} to Ω_{rc} again, using interpolation (4.4).

$$\Omega_{ex} \mapsto_{t_{idx}=0}^{e_j^{'}} \Omega_{rc} = e_j^{''} \tag{4.4}$$

For each known time index from the space-time archive (called a capsule in Fig.4.14), the boundary conditions are prescribed around the window by interpolating the $e_j^{'}$ boundary fields from the $\Gamma^{'}$ locations, to the new $\Gamma^{''} = \partial D^{'}$ as a subset of Ω_{rc}.

There is no other way for the 3D subdomain to be aware of what happens outside its universe. For greater efficiency, the prescribed boundary values

can be also interpolated for unknown time indexes, if the time resolution inside the capsule does not match the one in the original simulation. The reconstruction of the internal fields, e_j'', is performed using the same solver that was deployed in the global simulation run.

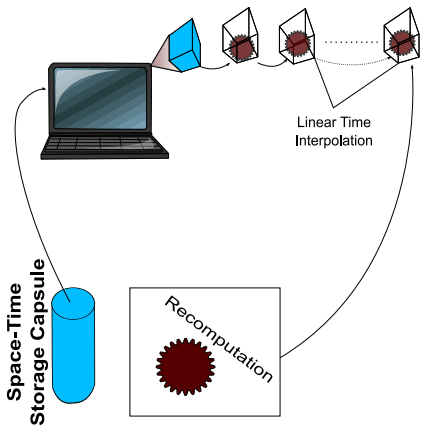


Figure 4.14: Solution A: Space-Time Window Reconstruction

The overall algorithm stages are presented in Fig.4.15. An extraction mesh is overlaid onto the global simulation in order to generate the geometric locations for field extraction.

The fields from the global simulation are then interpolated at the extraction locations, in order to be able later to form complete initial conditions and

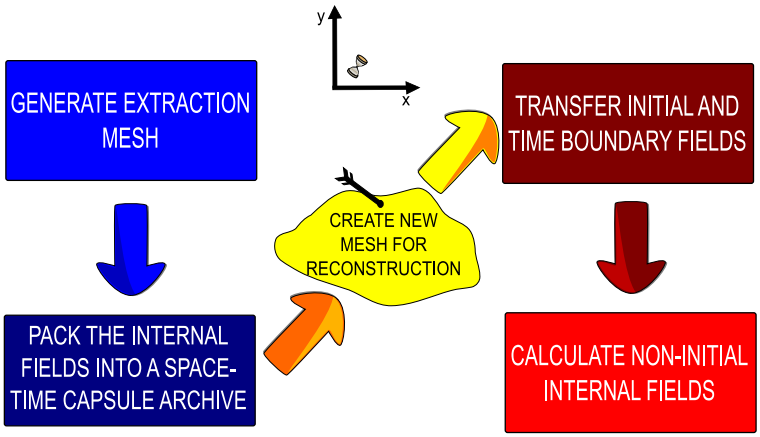


Figure 4.15: Main Stages for the Space Time Window Method

boundary prescriptions, in a space-time capsule archive. A new mesh is created for the reconstruction process, and the field values from the space-time capsule are used to approximate the initial conditions and the boundary evolutions of the space-time window. The missing internal fields are then recalculated using the same numerical solver.

The space-time window is, in fact, a transient submodelling window. However, since it involves unsteady turbulent phenomena at the boundaries, the procedure is very difficult to implement and much more complex than what is understood by submodelling, in the literature [15]. One may obtain the same flow features, but never the very same fields. If the very same fields are required, the next section provides a proper solution.

4.4 Solution B: Space-time window reconstruction based on interprocessor traffic

The previous solution allows for the reconstruction of the flow features, but not for the very same floating point numbers. In order to obtain the very same floating point numbers, Solution B proposes that the interprocessor traffic is intercepted and archived – Fig.4.16 describes this scenario.

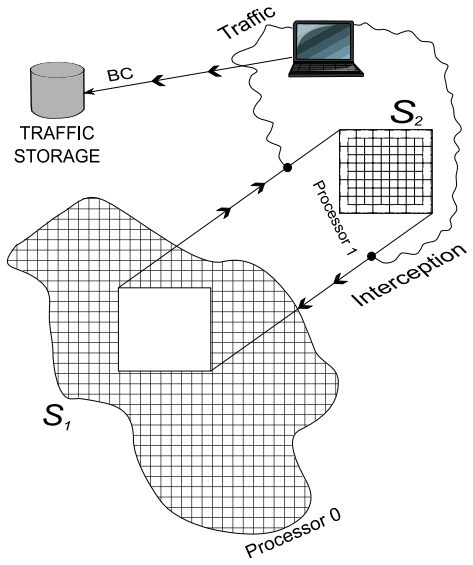


Figure 4.16: Solution B: Interception and Storage of Interprocessor Traffic

In order to obtain a space-time window, manual domain decomposition is used to split the global analysis domain D into two subdomains, S_1 and S_2. The amount of parallel interprocessor traffic generated by the two processors

associated with the subdomains strictly depends on the convergence of the solution.

On the other hand, the smaller the ΔT time step, the higher the number of iterations and traffic volume. With this method, skipping time steps is not possible at all. To overcome the very strict limitations, one needs to make sure that S_2 has a much finer mesh than S_1.

A polyhedral cell consists of points, edges and faces. If Ω_2 is the mesh of S_2, and Ω_1 the mesh of S_1, then the common boundary between S_1 and S_2 is defined by $\partial\Omega_2$, and consists of polygonal faces. Let the boundary cells of S_2 and B_2, be all the polyhedra in Ω_2 which contain boundary connection faces with Ω_1. Then, the internal cells of Ω_2 are defined by $I_2 = \Omega_2 - B_2$.

This method is applicable when I_2 can be refined such that the traffic between the two processors requires less storage then the actual fields in Ω_2.

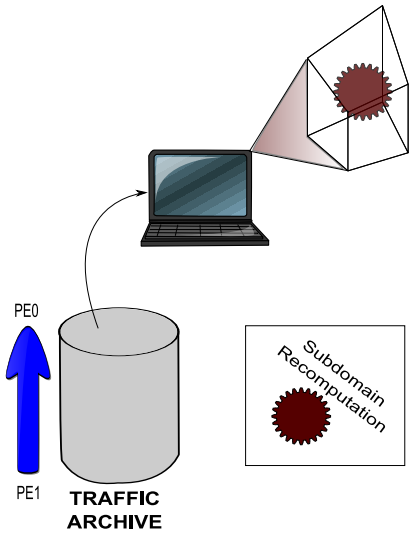


Figure 4.17: Solution B: Space-Time Window Reconstruction

The traffic is deposited into the traffic archive – Fig.4.17. When S_2 needs to be reconstructed, the solver reads the traffic archive and uses the commodity power of end-user hardware to recalculate the solution on Ω_2.

This method guarantees that the very same floating point numbers are obtained, but does not provide any more flexibility to the user than the usual submesh extraction techniques. Therefore, the two implementations are complementary to one another, and provide a functional way, for actually using the 'space-time window reconstruction' concept in practice. The next section synthesizes these ideas as proposed solutions to the limitations in the state of the art.

4.5 Closure

This chapter introduced the concept of space-time window reconstruction as a solution for the state of the art limitations, with a precise purpose of alleviating the bottlenecks described in the problem statement.

Both Solution A and Solution B are very complex and require gradual validations. Solution B is a particular version of Solution A, where no sources of errors are present. Solution A provides more flexibility and independence for the user, but trades-off accuracy and may require revalidation of the physical phenomena. The second solution, on the other hand, is just as flexible as the standard submesh extraction techniques, but guarantees the accuracy of the reconstruction and solves the data deluge problems.

In order to fully comprehend the necessary mechanisms and the constraints that need to be satisfied, a strategic approach to the problem is necessary. The next chapter introduces the strategy that was used to develop the 'space-time window reconstruction' concept, layered on levels of difficulty.

'Make my enemy brave and strong, so that if defeated, I will not be ashamed.'

The Pima people

5

Research strategy for space-time window reconstruction in CFD

The new concept of space-time window reconstruction is complex, of high impact, and involves a large spectrum of scientific fields.

The author considers that such a demanding goal can only be accomplished in gradual difficulty levels, starting up with simple but solid foundations and continuing with more and more complex levels on top. Since the application is for CFD, the complexity levels need to be organised based on CFD specifications.

As any other engineering field, computational fluid dynamics is an art of approximation. There are three major types of approximation in CFD, according to Muntean [119]:

- Temporal approximation, which implies the estimation of time-dependant variables and coefficients, and the selection of the ΔT time step range.
- Spatial approximation, which defines the number of spatial variables used in the model.
- Dynamic approximation, which selects the most relevant components from the equation models that best describe the physical system at hand. Terms which have a negligible impact are removed, so that the computational resources are exploited to the maximum.

During the temporal approximation, the size of ΔT is decided depending on the numerics involved, and how fast the selected variables and coefficients evolve in time.

In spatial approximation, it may be decided that unidimensional (1D), bidimensional (2D), cvasi-3D or complete tridimensional models are going to be used. This depends on the physical phenomena and the available input data.

With this knowledge at hand, the thesis research is organised into four different stages. At the very first level, a non-dimensional, laminar (non-turbulent) steady-state flow of an ideal fluid, is used, taken from [109]. It is called a 'proof of concept' in Fig.5.1, and it was designed with simple FEM techniques.

The simplicity of the case translated into a single system of linear equations in matrix form, and 2D interpolation with triangular meshes. By the nature of the FEM, the field values were present at nodal points in the mesh, obtained via triangulation. The only field in the simulation was non-dimensional.

After the proof of concept was successful, the complexity was increased by adding rotationary phenomena, and switching to the FVM. The number of spatial variables was increased from 0 to 2 in the cvasi-3D test case. In both cases, the space-time window was positioned to start from known initial conditions, identical to those in the global simulation.

The FVM mesh moved the field values from the nodal points to the centre of the polyhedral cells, and face values had to be interpolated.

With a higher number of spatial variables, vector fields now had to be dealt with using their unit versors. The number of fields and field types increased and each had to be handled independently.

Because of turbulence, the velocity fields capture spinning vortices that go into and out from the space-time window.

At the next level, a well known ERCOFTAC benchmark case, introduced by Lyn and Rodi [120], is used to check if the method is functional. The equations are elaborated by Anton and Baya [121]. The simulation is now complex LES with unsteady turbulence features, and very fine mesh resolution and small time steps. The supercomputers in Stuttgart were used to develop this test case [1].

To make it more practical, the space-time window was positioned after 5 seconds of global simulation, and the initial conditions had to be derived geometrically, both during the extraction (packing) phase, and during the transfer of the fields onto the reconstruction mesh. This raised an incredible amount of floating point issues, as only someone who experienced the problems can understand. Floating point numbers are known to be inexact and problematic when one has to program boolean operations depending on them. Computational geometry logic, like in this case, is a good example of how bad things can get.

Time is now actually part of the simulation, using a backward differencing scheme. This means that all the errors are cumulative during each iteration, and that a new system of equations is assembled for each one of the time steps. Information that is necessary to approximate the initial conditions and to prescribe time-varying boundary conditions had to be organised in a space-

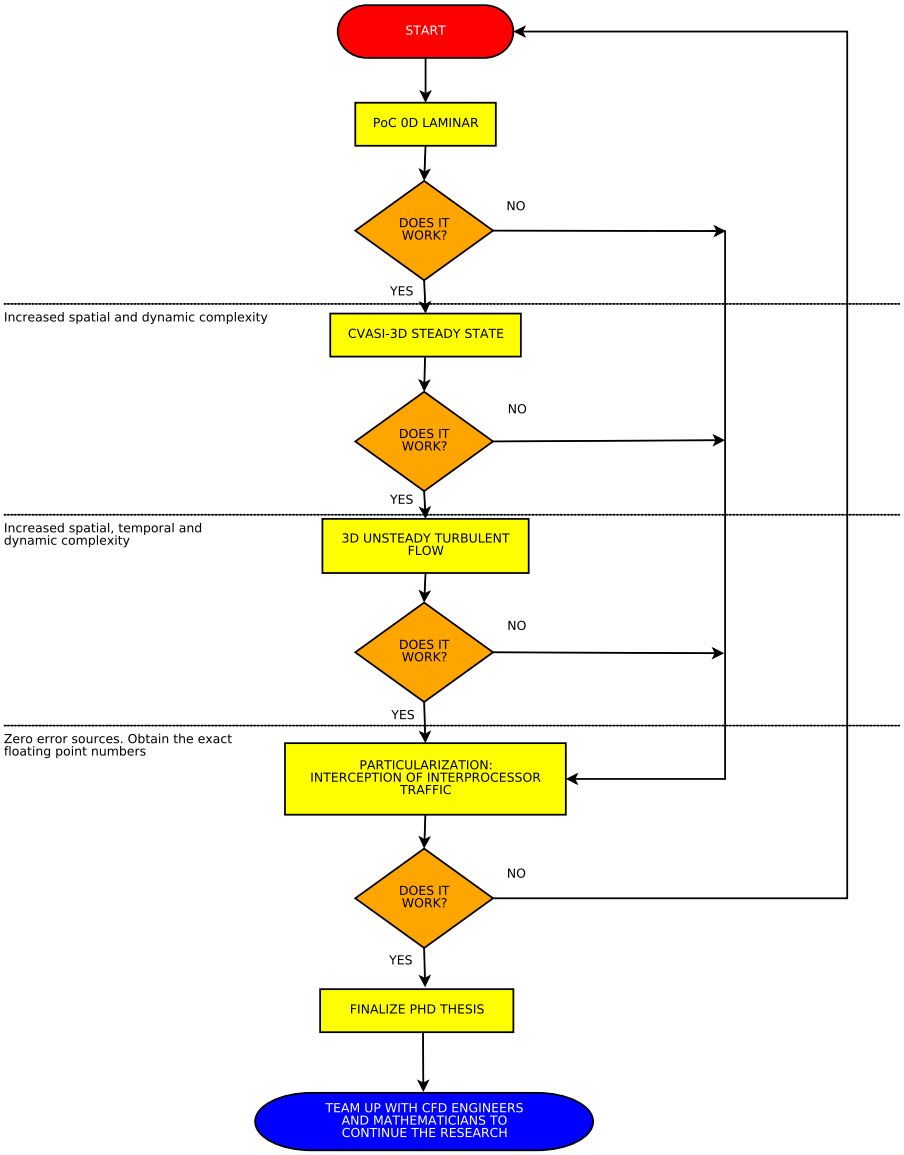


Figure 5.1: Research Strategy for Space-Time Window Reconstruction

time format, and then used correctly, in order to configure the reconstruction test cases.

With the method also successful at this level, the author attempted to obtain the very same floating point numbers during the reconstruction, which led to the development of the traffic interception approach. This approach has also been envisioned from the beginning as a backup resort if everything

else fails, but without any fail-safe guarantees. The author had absolutely no idea if any of the attempts were going to be functional when he started to work on the problem, and this can be easily spotted from the strategy in Fig.5.1.

This time the approach is radically different. Instead of using network tools to intercept the traffic, the author wanted to measure the exact number of floating point values transported through the processor patches. For that, the processor patch implementation was modified allowing for algebraic-level information count, far away from the jungle of communication protocols and headers. It was quite a surprise to see that infinitely small approximations in the traffic data, like truncations of the number of decimals to 10^{-300}, completely destroy the recalculation process. This clearly made the point that no matter how small, any accuracy error will make it impossible to obtain the very same floating point numbers for the finite fields. However, obtaining the same feature flows has proven to be possible, even when interpolation errors are cumulatively introduced.

The next part of the thesis will show the results obtained during each level of complexity, and the lessons learned from each of the test cases.

6

Finite element proof of concept. Field reconstruction inside a window subdomain of a 2D steady flow

As shown in Chapter 5, this proof of concept is a non-dimensional, laminar (non-turbulent) steady-state flow of an ideal fluid, modelled using the FEM. This is the simplest scenario driven to test if the space-time window concept can be applied to CFD, and eventually understand how. The results have been published in Anton [122] and Anton and Creţu [123].

A non-dimensional simulation, according to Muntean [119], means that there are no spatial or temporal variables. The U_x and U_y velocities of the flow can be derived from the non-dimensional stream function, described in the next section. The stream function is explained by Resiga [124].

The simplicity of the test case allows the focus to concentrate on computer science issues, in order to define the basic mechanisms and issues to be taken into account for space-time window reconstruction.

The source code and the test case are fully described by Resiga et al. [109] and are based on the PETSc Toolkit from Balay et al. [26].

6.1 The test case

For FEM, the unknown variables are located at the vertices of the elemental shapes, and boundary conditions are set at nodal points.

The stream function ψ is defined by equation (6.1).

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0 \tag{6.1}$$

Given that the stream function is known, the horizontal and vertical velocities, U_x and U_y, are produced by equation (6.2).

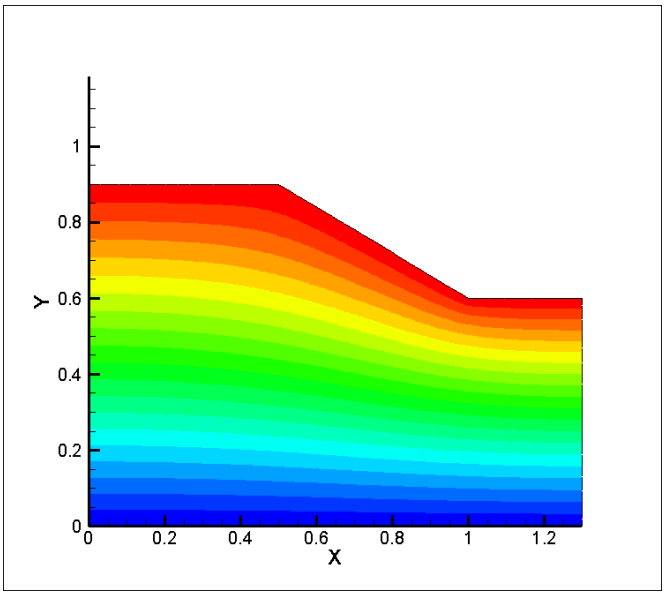


Figure 6.1: Streamlines for the global simulation

$$U_x = \frac{\partial \psi}{\partial y}, U_y = -\frac{\partial \psi}{\partial x} \tag{6.2}$$

Fig.6.1 shows the calculated streamlines for the global simulation run.

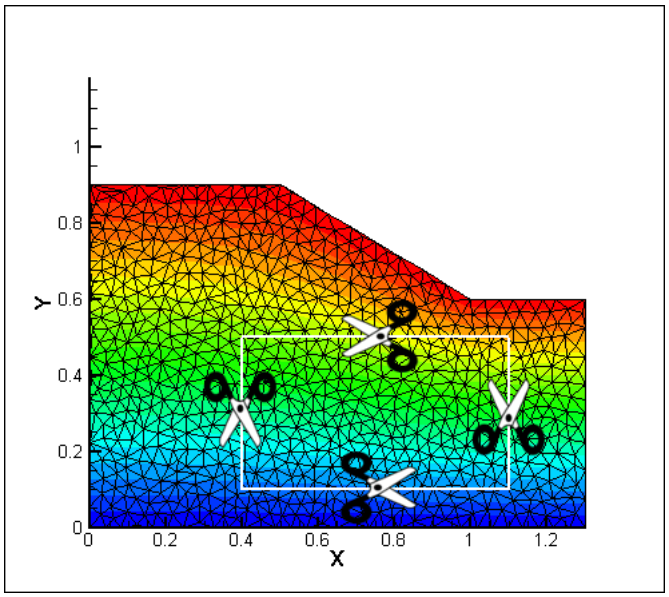


Figure 6.2: Extraction at the Border of the Rectangle

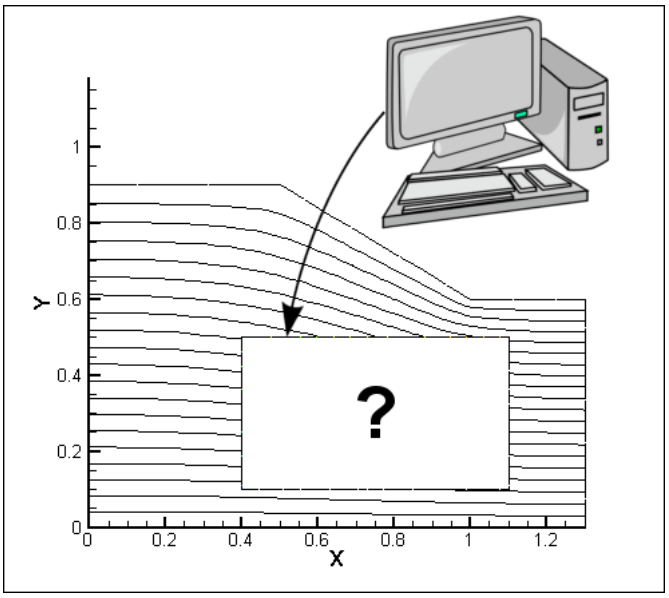


Figure 6.3: Configuration of Boundary Values

The ψ value is a scalar with values between $\psi \in [0, 1]$.

Fig.6.2 shows the extraction process where the Γ border of a window defined by coordinates (0.4, 0.1) (1.1, 0.5) is cut out and archived.

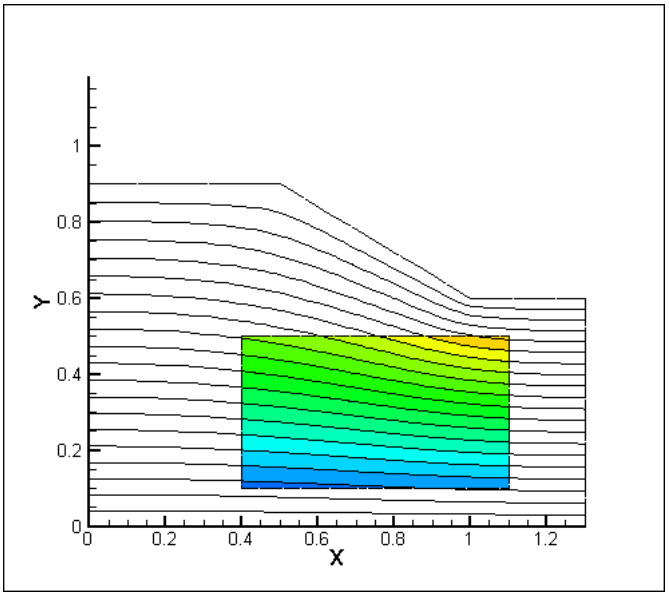


Figure 6.4: Data Inside the Rectangle Border is Reconstructed by Computation

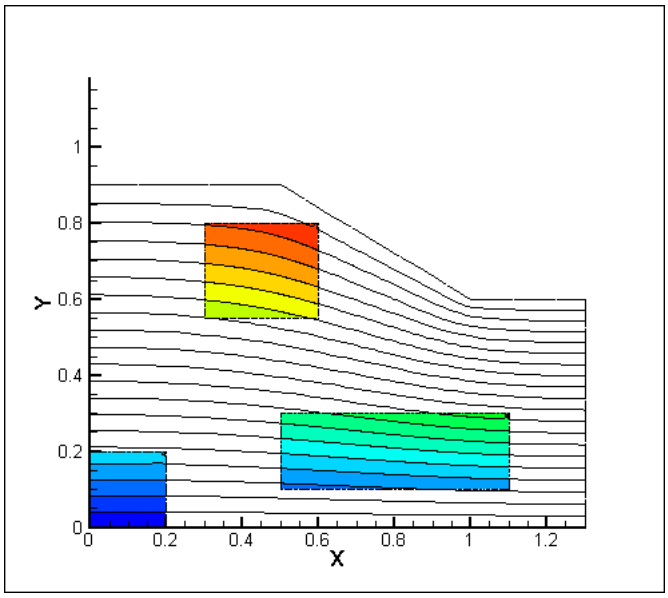


Figure 6.5: The Proof of Concept Works

For this particular case, the ψ value is extracted at a number of 10000 points, equally distributed across the 4 border segments of the region.

The original mesh contains 810 vertices defining 1556 triangular elements. The reconstruction mesh is defined by 2268 vertices and 4427 elements. The Triangle library is used for mesh generation [125].

In order to match the new, reconstruction mesh with the original sample set linear interpolation is used – Fig.6.3

The data within the extracted region is fully reconstructed based solely on the border values, as shown in Fig.6.4. The flow streamlines clearly resemble the original solution without disruption. The absolute accuracy obtained by comparing values at different random points within the reconstructed domain is $\epsilon = 10^{-6}$ against the original solution.

Other windows with coordinates of $(0,0)(0.2,0.2)$, $(0.5,0.1)(1.1,0.3)$ and $(0.3,0.5)(0.6,0.8)$, are reconstructed in Fig.6.5 with the same level of accuracy.

The same experiment is repeated with only 100 extraction points, and the same level of accuracy is preserved. The storage requirements necessary for archiving the ψ values in 100 points represent only 6.05% of the original data.

A particular case for predicting the ψ function without computation can be implemented by replacing the geometry in Fig.6.1 with a 1-by-1 square, called $D^{'}$. Let the boundary conditions be defined by $\Gamma^{'} = \Gamma^{'}_{upper} \cup \Gamma^{'}_{lower} \cup \Gamma^{'}_{left} \cup \Gamma^{'}_{right}$.

Let the boundary conditions for $\Gamma^{'}_{lower} = 0$ and $\Gamma^{'}_{upper} = 1$ be of Dirichlet

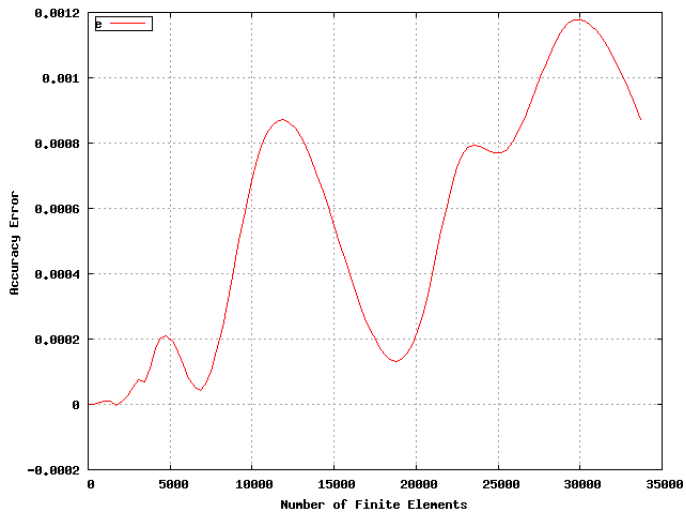


Figure 6.6: The accuracy ϵ varies when the FEM Resolution is Increased

type and the $\partial\Gamma^{'}_{left} = \partial\Gamma^{'}_{right} = 0$ be of Neumann type, for the inlet and the outlet of the test analysis domain. If the derivative of a function is 0, the function is constant over the set of corresponding boundary points; the stream function can be traced to (6.3) for the 1-by-1 square geometry.

$$\psi^{'}(x, y) = y \tag{6.3}$$

In order to define the accuracy of the reconstruction, let a discrete subset of $D^{'}$, be defined by $Q = \{(x, y)_1, \ldots, (x, y)_n\} \subset D^{'}$. If $\psi_o(x, y)$ and $\psi_r(x, y)$ correspond to the original and the reconstructed values of the stream function at Euclidean coordinates (x, y), then the accuracy of the reconstruction, ϵ, is defined by (6.4).

$$\epsilon = |\psi_o(x, y) - \psi_r(x, y)| \tag{6.4}$$

The number of arbitrary test points is given by n. In order to test the accuracy it is important to test both arbitrary points, and also the previously computed locations used by the original simulation.

Define the interpolation resolution $R_i^{'}$ as the cardinality of the $\Gamma^{'}$ set, or $R_i^{'} = |\Gamma^{'}|$. The interpolation algorithms are defined by the set $A = \{$Linear, Polynomial, Cubic_Spline, Cubic_Spline_Periodic, Akima, Akima_Periodic$\}$ based on the implementations in [126].

The original resolution $R_o^{'}$ and the reconstruction resolution $R_r^{'}$ are defined by the number of elemental shapes contained inside the user-defined region.

In Fig.6.6 it is shown that the accuracy, ϵ, compared against the value of

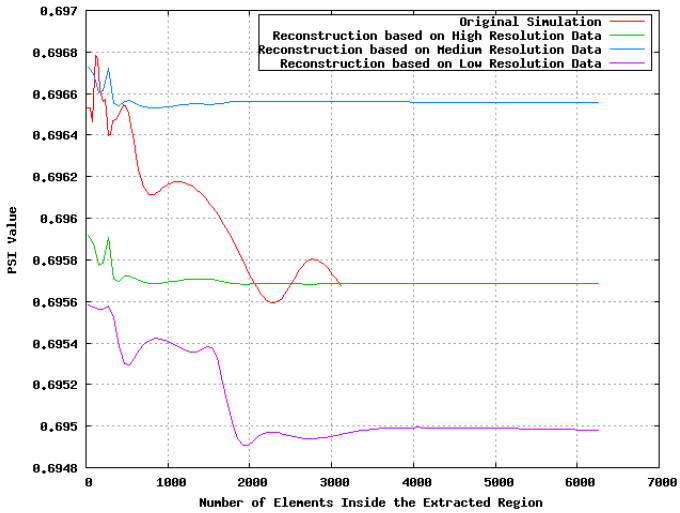


Figure 6.7: Reconstruction based on Different Original Mesh Resolutions

the estimated $\psi^{'}$ function, is increasing with the $R_o^{'}$ resolution of the original mesh. This leads to the conclusion that it is very difficult to compare two or more numerical simulations against each other, in terms of retained accuracy, without referencing measurement information. For this particular case, it appears that coarser mesh resolutions of $R_o^{'} = 18 - 25$ elements are superior in terms of accuracy to the finer meshes.

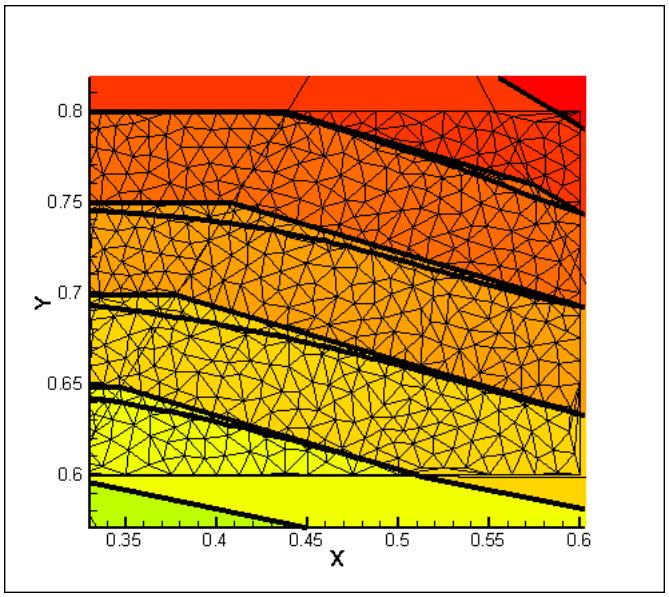


Figure 6.8: Accuracy of Fine Reconstruction based on Coarse Simulation

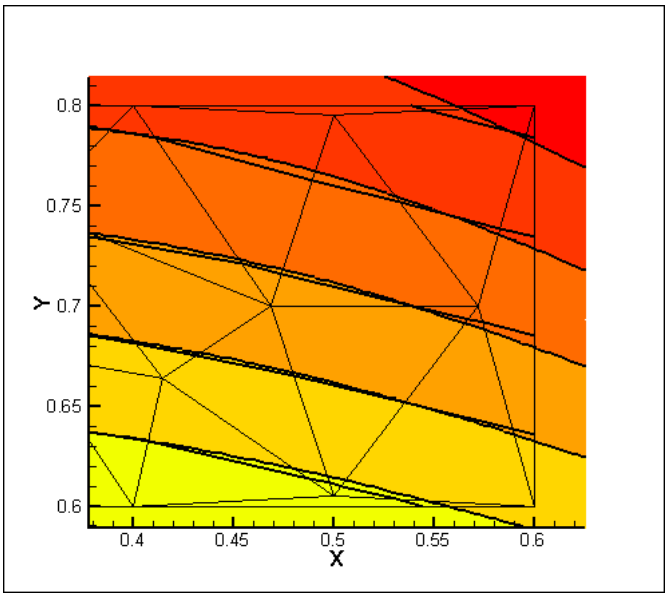


Figure 6.9: Accuracy of Coarse Reconstruction based on Fine Simulation

In Fig.6.7, the $\psi^{'}$ value is obtained by starting with data extracted at different original $R_o^{'}$ mesh resolutions.

Clearly the reconstruction based on finer original simulation intersects the original solution in multiple places. The reconstruction based on a very coarse original simulation mesh fails to match any of the initial values, and the reconstruction based on medium mesh simulation resembles the parent solution at a coarser reconstruction level.

If the initial $R_o^{'}$ is not too coarse, the reconstruction preserves the original accuracy even when computed with fewer elemental shapes. On the other hand, if the initial solution is not fine enough, the reconstruction will fail.

Fig.6.8 shows what happens when the original simulation is coarse, and the reconstructed region is finer. The flow streams do not coincide any more because the extraction points are based on far less accurate data, as in the low profile from Fig.6.7.

In Fig.6.9, the original simulation is very fine, but the reconstruction is too coarse. Again, the streamlines do not coincide any more, as shown in Fig.6.7 at the high profile, with a starting region of 18-25 elements inside the extraction window.

The influence of $R_o^{'}$ and $R_r^{'}$ have been studied during the reconstruction phase. Fig.6.10, shows the results obtained with different interpolation algorithms used to configure boundary conditions between the extracted $\Gamma^{'}$ set of points. For this, a $|\Gamma^{'}| = 50$ points is used.

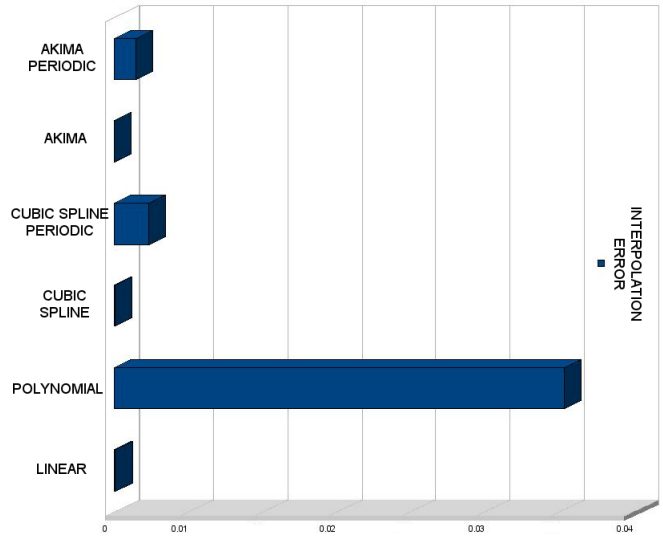


Figure 6.10: Error Introduced by Interpolation Algorithms

The tested algorithms are implemented by the GNU Scientific Library package [126]. The results for cubic splines, Akima and linear interpolation are very good. Polynomial interpolation fails to correctly provide the missing values for larger $R_i^{'}$ resolutions. This happens because of the Runge oscillatory effect, known for the higher grade polynomials [127].

In order to study the Runge effect, Fig.6.11 shows that the values interpolated by the simple polynomial algorithm start to oscillate when the

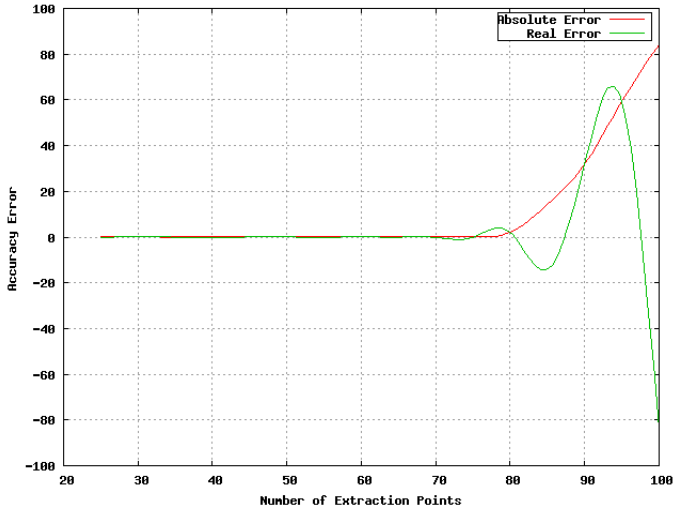


Figure 6.11: Runge Effect on Polynomial Interpolation Points

cardinality of Γ is incremented; the number of extraction points define the grade of the interpolation polynomial and break out of the bounds when higher than 70. Polynomial interpolation works well for low number of R_i' points and is easy to implement.

All the other algorithms work well regardless of the R_i' resolution, and scale well to a larger set of extraction points. The interpolation stage is responsible for reproducing the boundary conditions correctly. If the interpolation algorithm fails, the reconstructed problem, within the space-time region, becomes entirely different than the original, and all the other parameters are useless. The following section drives the first remarks that could be concluded around the idea of space-time window reconstruction in computational fluid dynamics.

6.2 Concluding remarks

This very simple CFD test case was designed in order to create a proof of concept for the proposed idea of space-time window reconstruction, and verify if the goals are even achievable. The test was a success, and the data was reduced to 6.05%. Multiple reconstruction windows were produced retaining the original flow features with 10^{-6} accuracy.

On the other hand, the lessons learned were harsh: even for the most simple fluid flows, the success depends on the resolution of the original domain and the reconstruction mesh. Experiments have shown that even here, without turbulence or rotationary flows, the method does not behave like classic submodelling – reconstruction based on a low number of extraction points, fails to reconstruct the fields and the flow features, no matter how fine the reconstruction mesh is.

The interpolation algorithm should always be the simplest one – linear interpolation. It appears that if linear interpolation fails, it is more proper to experiment with mesh and extraction resolutions rather than change the interpolation scheme. The next text case will introduce rotationary flows into the scene.

7

Finite volume test case. Field reconstruction inside a window subdomain of a cvasi-3D steady flow

The purpose of this experiment is to check if the method can be applied on turbulent flows, and mainly on turbulent boundaries for the cut region. Due to the nature of the FVM implementation in OpenFOAM [27], the 2D case has to be rendered in 3D space with only one discretisation level for the Z-axis. This is called 'cvasi-3D', as shown in Chapter 5. The GSL library [126] is still used for interpolation, but this time the extraction of the fields is performed with internal OpenFOAM methods.

For the finite volume method, the unknown variables are centred inside the elemental volumes, and boundary conditions are configured as patches at the surface level. The results have been published in [122]. The chapter will now continue with the presentation of the test case, and the results obtained at this stage.

7.1 The test case

In Fig.7.1 the test case is taken from [128] and standard OpenFOAM [27] tutorials. It is a steady rotationary flow over a backward-facing step.

The governing equations are given by (7.1) and (7.2).

$$\nabla \cdot U = 0 \tag{7.1}$$

$$\nabla \cdot (UU) + \nabla \cdot R = -\nabla p \tag{7.2}$$

where U is the velocity, p is pressure and R is a viscous stress term.

The global simulation in Fig.7.2 shows the velocity contours at time step $t_{idx} = 1000$. The time steps in this simulation do not represent real, physical

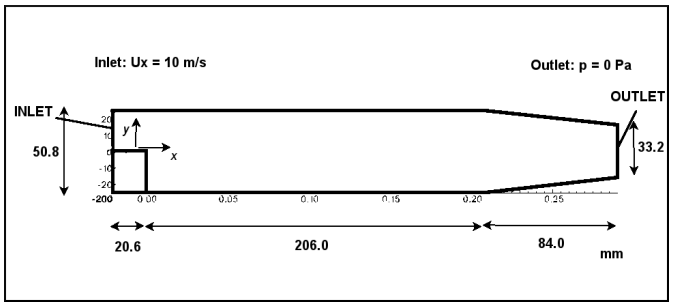


Figure 7.1: Geometry of a Backward-Facing Step

time, they are just the means of achieving convergence. Time in a steady-state simulation, if present, is called 'pseudo-time'.

A user-defined rectangular hexahedron with coordinates (0.05 -20 -0.5) and (0.2 10 0.5) is extracted from the solution in Fig.7.2. Each side of the 2D rectangular projection contains 100 points. After sampling the values for p, U_x, U_y, U_z and the necessary fields, the values are interpolated using cubic splines, as in Fig.7.3.

The splines are used to configure the values at the boundary of the reconstruction domain: smartPatches are setup around the window (in this case, a cvasi-3D hexahedron) and the same solver is used to recompute the internal fields.

Fig.7.4 shows that the reconstructed vector fields retain the original flow features and depicts the velocity glyphs forming a vortex.

A zoom-in, Fig.7.5, shows that the vortex ends around $x = 0.15$ as the stream continues to follow a forward path across the geometry.

The user-defined region of interest at iteration (pseudo-time step) 1000

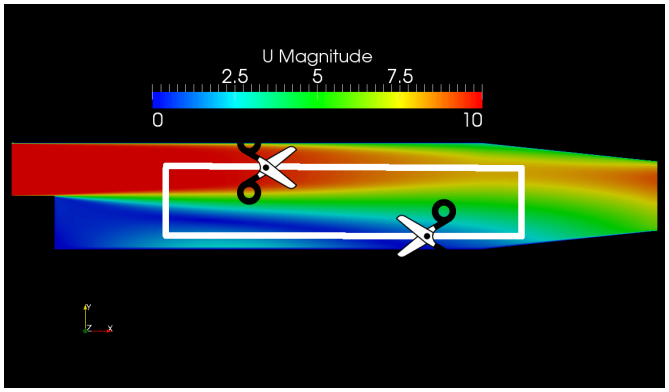


Figure 7.2: Global Simulation

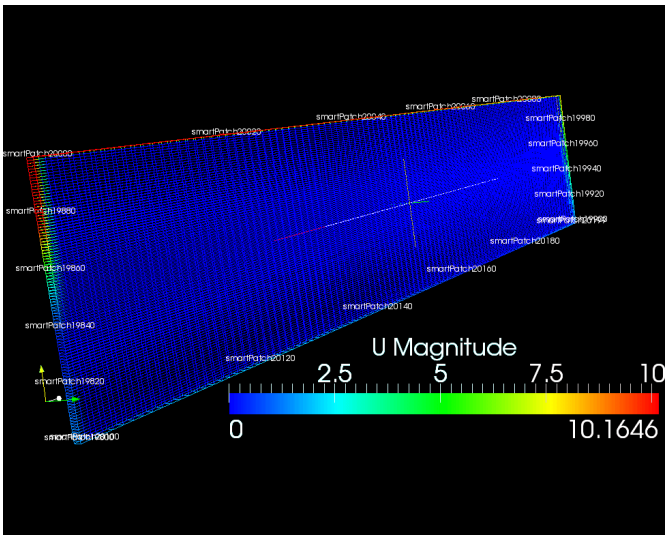


Figure 7.3: Interpolated smartPatches around the Reconstruction Window

reconstructs the original pressure and vector fields with an absolute accuracy of $\epsilon = 10^{-1}$. This was checked by sampling arbitrary points inside the reconstructed fields and comparing them with the values for the same locations in the global simulation.

The accuracy of the reconstruction occasionally breaks up to $\epsilon = 0.11$. Fig.7.6 shows the evolution of a random pressure probe inside the analysis

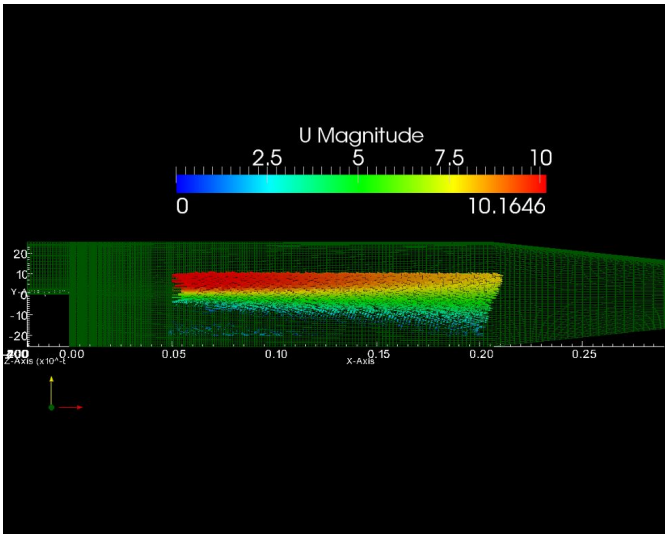


Figure 7.4: Data Inside the Window is Reconstructed

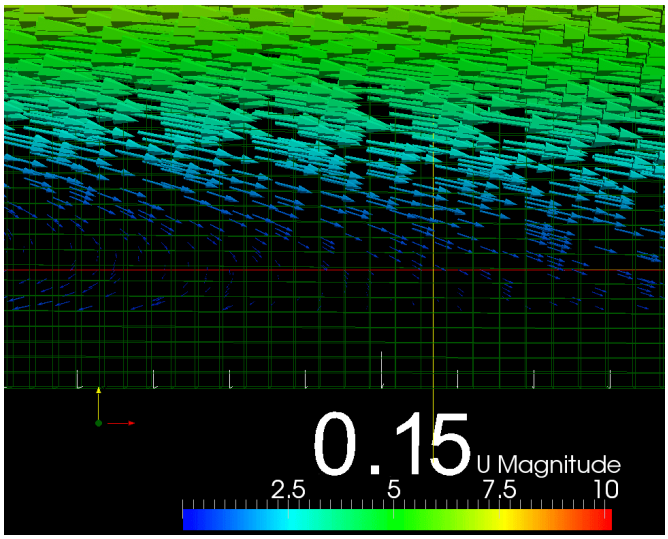


Figure 7.5: Zoom-in on Reconstructed Area

domain, during each iteration. At first, during the initial conditions, the values are identical. The first iteration breaks them apart, and then they converge together to be indistinguishably identical after less than 1000 rounds.

The original mesh consists of 12225 hexahedral cells, 49180 faces and 25012 points. The reconstruction mesh has 10000 hexahedra, 40200 faces and 20402 points. The storage requirements are reduced to 0.27% per time step (consisting of the extracted field values), with the original geometry adding another 10.4% from the original simulation data.

It is important to note that in this cvasi-3D case time is only pseudo-time, because of the steady-state model. That makes time steps independent of one

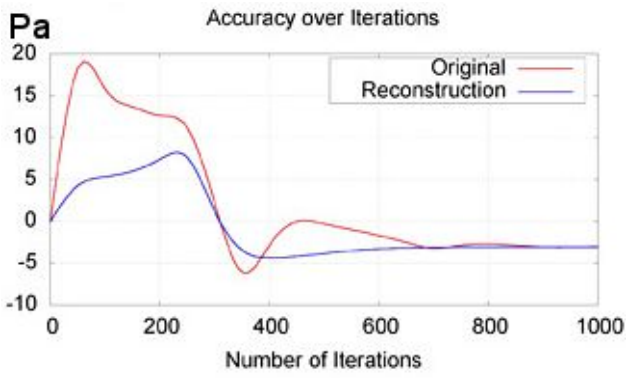


Figure 7.6: Pressure probe convergence

another, and easy to reconstruct. For transient simulations, the influence of the neighbouring time steps is critical, as it will be shown in the next chapters. The conclusions observe that even for rotationary flows, the space-time window reconstruction idea is feasible.

7.2 Concluding remarks

This test case introduced the space-time window reconstruction idea to rotationary flows. Basically, it confirms that when having turbulence at the boundary conditions, with vortices that enter and exit throughout the cut region, the reconstructed fields still preserve the flow features. Even more, this validation provides the basic concepts and mechanisms that need to be developed in order to obtain similar results with unsteady flows in OpenFOAM.

The next chapter presents two test cases for unsteady turbulent flows, and their reconstruction results obtained with OpenFOAM.

‘A man must make his own arrows’.

Winnebago

8

Space-time window reconstruction in a 3D unsteady complex flow based on submodelling

The previous test cases used implementations of lower complexity. As shown in the concentrated schema of Chapter 5, this chapter uses fully transient (unsteady) turbulent flows, to prove the space-time window reconstruction concept. It is organised into four sections: one that explains the implementation, one that discusses how and when time interpolation can be introduced, another one with a very complex ERCOFTAC demonstration – as a means for applying the new space-time window reconstruction concept on real-world, well-known computational fluid dynamics test cases – and the closure with the concluding remarks. The chapter is now continuing with implementation details.

8.1 Implementation details

The ‘Extend Project’ for OpenFOAM is an independent, community driven toolbox, improved with user contributions, and developed in the spirit of the Open Source movement. Considering that the space-time window reconstruction concept is new, one goal of the research is to produce a new tool, suited for the space-time window reconstruction concept and capable of generating new scientific results. Right now, the tool plays the role of a ‘software booth’, which needs to be improved with the help of professionals coming from other fields of expertise, such as hydraulic engineering and applied mathematics. As a medium-term prospect, after all the validations which are not computer science related are performed, the author intends to submit the final revision of the software as an independent tool for the ‘Extend Project’ community.

Because the complexity of the test-cases introduced in this chapter reaches the highest level, this section continues with specific implementation details and the approaches chosen to develop a functional solution.

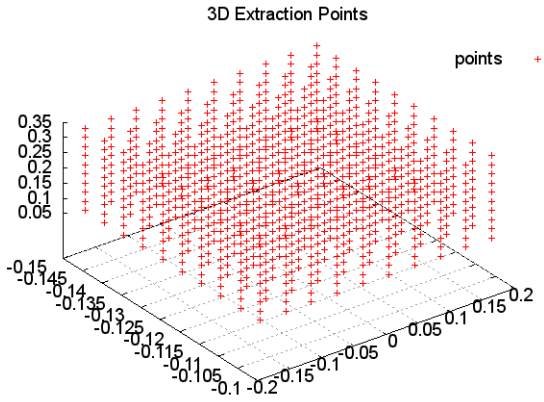


Figure 8.1: Extraction Points for the Initial Fields – $\Gamma_{initial}$

One of the most serious issues when implementing boolean logic is the handling of decisions when they have to be based on floating point numbers. Floating point arithmetic only retains data with limited accuracy, and can not always represent the exact numbers that are supposed to be carried by the hardware [129]. This problem is negligible in terms of solution accuracy, but it turns out to be a nightmare when the program flow depends on floating point comparisons. Results have been previously published in Anton [130], and Anton and Creţu [131].

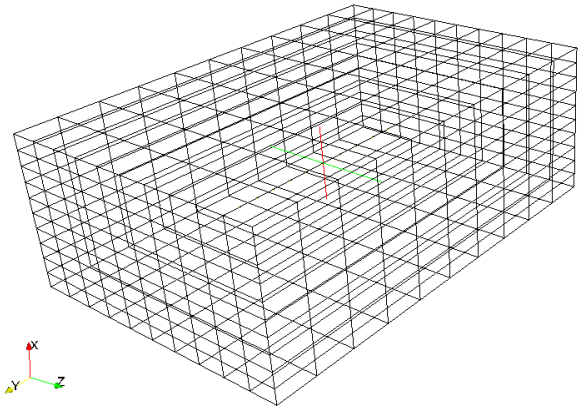


Figure 8.2: Subdomain Reconstruction Mesh

The easiest way to stumble into the floating point jungle is to implement, or use, high resolution geometric algorithms. A common example from numerical simulation software, can be related to volumetric structures, with point location algorithms. In other words, trying to locate a geometric point, on the surface of a control volume (cell), or on the line belonging to an edge, can get very tricky. The usual way to go around is by range-checking. The

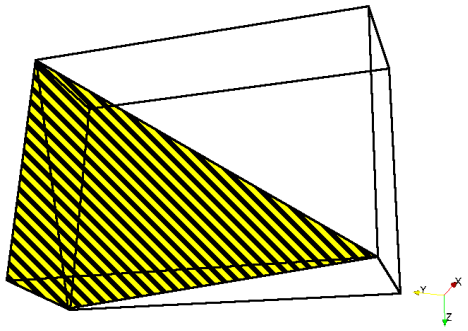


Figure 8.3: Tetrahedral Decomposition

developer can define whatever seems to be small enough, small, very small and even smaller – and sometimes it works, sometimes it doesn't. And sometimes the same goes for the upper limits! The point is, the problem usually surfaces only when it is too late, in the most unpleasant moments.

A smarter, and sometimes useful method for dealing with boolean geometric statements, that have to rely on floating point values, is to use complementary, and redundant, geometric algorithms for cross-validation. Or, one could use a different type of coordinate system to try and obtain the same program flow. This basically consumes resources and slows down both the implementation, and the running time of the program.

This thesis certainly does not try to explain floating point development techniques – a very good review on the subject is available in [132]. On the other hand, this kind of circumstances may be encountered millions, billions, or even trillions of times, for instance when sampling field data inside moving meshes. Smarter approaches are always necessary, to get things going. For our test-cases, point location problems were performed millions of times, with static meshes.

Consider the $\Gamma_{initial}$ set of points in Fig.8.1. The work has been published in Anton and Creţu [131]. The implementation is based on the finite volume, OpenFOAM toolkit [27]; phenomena evolution inside the finite volume cells is considered to be linear.

The scalar and vector fields used by the numerical solver are interpolated

at the coordinates provided by the $\Gamma = \Gamma_{initial}$ set for the initial fields, and by $\Gamma = \Gamma_{limit}$ for the subdomain boundary faces.

A new mesh is generated inside the subdomain, in order to reconstruct the fields by recomputation – Fig.8.2. To transfer the initial fields to the new mesh, the $\Gamma_{initial}$ points are considered to describe a regular, hexahedral extraction mesh, similar to the one in Fig.8.2, and usually on a different resolution. For practical reasons, any space-time window is considered to be a hexahedron, defined by the coordinates of a bounding box. This extraction mesh is decomposed into tetrahedral elements, like it is shown with the cell in Fig.8.3.

```
a = x1 - x4;  d = y1 - y4;  g = z1 - z4;
b = x2 - x4;  e = y2 - y4;  h = z2 - z4;
c = x3 - x4;  f = y3 - y4;  k = z3 - z4;

A = e*k - f*h; D = c*h - b*k; G = b*f - c*e;
B = f*g - d*k; E = a*k - c*g; H = c*d - a*f;
C = d*h - e*g; F = b*g - a*h; K = a*e - b*d;

Z = a*(e*k-f*h) + b*(f*g-k*d) + c*(d*h-e*g);

l1 = (1/Z) * (A*(x-x4) + D*(y-y4) + G*(z-z4));
l2 = (1/Z) * (B*(x-x4) + E*(y-y4) + H*(z-z4));
l3 = (1/Z) * (C*(x-x4) + F*(y-y4) + K*(z-z4));
l4 = 1.0 - l1 - l2 - l3;
if (l1 + l2 + l3 == 1.0) l4 = 0; // FP limitation

if ((l1>=0)&&(l1<=1) && (l2>=0)&&(l2<=1) &&
    (l3>=0)&&(l3<=1) && (l4>=0)&&(l4<=1)) {
        pval = l1*val[4] + l2*val[5] + l3*val[0] + l4*val[7];
        return (pval);
}
```

Figure 8.4: Tetrahedral Interpolation Algorithm

Equations (8.1), (8.2) and (8.3) show the formulae for tetrahedral interpolation. For each point $P(x, y, z)$ with the Cartesian coordinate vector p, the value of a field in point P is given by Φ_{3D}, where $\Phi_{i=1..4}$ is the value of the field in the vertices of the tetrahedra, and $p_{i=1..4}$ are the coordinates of the tetrahedra.

The λ_1, λ_2, λ_3 and λ_4 coefficients are the barycentric coordinates of the tetrahedra.

$$T = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix} \tag{8.1}$$

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = T^{-1}(p - p_4) \tag{8.2}$$

$$\Phi_{3D} = \lambda_1\Phi_1 + \lambda_2\Phi_2 + \lambda_3\Phi_3 + \lambda_4\Phi_4 \tag{8.3}$$

The 3D interpolation procedure is also described in Fig.8.4. Due to limited floating point representation and errors, it is necessary to perform redundant checks whenever possible.

The Γ_{limit} boundary conditions are approached with similar algorithms in 2D. The 6 surfaces of the extraction subdomain are decomposed into triangles, and each point $P(x, y)$ from the boundary is prescribed on the reconstruction mesh using the formula in (8.4).

A is the area of the triangle in Fig.8.5, described by Cartesian points $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ and $P_3(x_3, y_3)$, and Φ_1, Φ_2, Φ_3 are the field values at the triangle vertices.

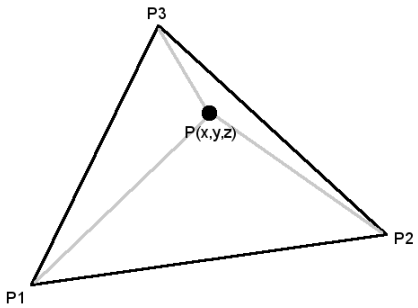


Figure 8.5: 2D Linear Interpolation inside Triangle

$$\begin{aligned} \Phi_{2D} &= \tfrac{1}{2A}((x_2y_3 - x_3y_2 + (y_2 - y_3)x + (x_3 - x_2)y)\Phi_1 \\ &+ (x_3y_1 - x_1y_3 + (y_3 - y_1)x + (x_1 - x_3)y)\Phi_2 \\ &+ (x_1y_2 - x_2y_1 + (y_1 - y_2)x + (x_2 - x_1)y)\Phi_3) \end{aligned} \tag{8.4}$$

For the vector fields, the 2D and 3D interpolation procedures are repeated for each of the versors.

The entire directory structure can be archived and compressed by any of the popular tools, for easier transportation.

The logic scheme of the extraction algorithm is presented in Fig. 8.6. For all the points in the extraction mesh (np), the first test verifies if the desired location is contained by the bounding box, (P in BB), containing the current source cell. If it is not contained, then it is clear that the present cell does not hold the coordinates.

Usual space-time windows can contain $10^3 - -10^5$ time steps, and require a great amount of iterations, in order to extract all the necessary points.

Having this in mind, irregular cell shapes and unstructured meshes can seriously lag the point inclusion tests.

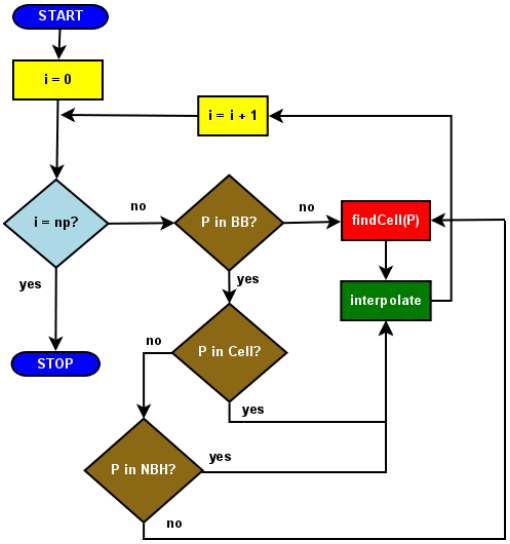


Figure 8.6: Point Extraction Logic

The previous location has provided the present cell, so the current location can only be one step away in the neighbourhood.

The finer the extraction mesh, the higher the probability that the desired location is somewhere in the adjacent layer, like in Fig. 8.7.

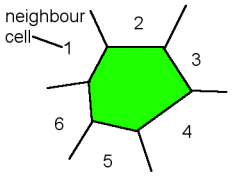


Figure 8.7: Neighbour Cells

If the extraction mesh is dense, there's also a chance that the desired location is still inside the present cell, and that the step value was not large

enough to take it outside the volume. Therefore, it is worthy to check for it with the P in Cell test.

For hexahedral meshes, the adjacent neighbourhood consists of only 6 cells. If the mesh is also regular, the inclusion tests can be performed faster, with simple arithmetic comparisons for minimal and maximal values. For very complex geometries, the algorithm can be extended, in order to include multiple layers within the neighbourhood.

The field values for the $\Gamma_{initial}$ and Γ_{limit} coordinates are stored in a subwindow directory like in Fig.8.8. Complete boundary values are saved for all the time steps.

Fig.8.9 shows the algorithm for the extraction mesh. The coordinates of each point are located at equal step distances, in all of the three dimensions.

```
./spaceTimeDir/
    5.000000e+00/full/scalars/k
    ..
    5.000000e+00/full/scalars/p
    5.000000e+00/full/vectors/U
    5.000000e+00/full/vectors/U_0
    5.000000e+00/scalars/k
    ..
    5.000000e+00/vectors/U_0
    ..
    5.005000e+00/scalars/k
    ..
    5.005000e+00/vectors/U_0
    boundingBox
```

Figure 8.8: Directory Structure for the Space-Time Window

The bounding-box check usually introduces additional computations inside the point iteration loop. The reason for this test is to deal with higher-order polyhedrons. Fig. 8.10 shows such an example.

The point-in-cell test is only performed if the location has been verified to be inside the bounding box. Otherwise, mesh.findCell() is going to be called, and the octree mesh structure will be scanned for the cell containing the new location. Much smarter algorithms can be developed based on these simple, yet elegant principles, also considering the time-varying nature of the data, and possible mesh movements. However, for the practical purposes, the goals have been achieved.

```
for (k=0; k<Ndensity; k++) {
        for (j=0; j<Cdensity; j++) {
                for (i=0; i<Rdensity; i++) {
                        origin = i*stepx + j*stepy + k*stepz;
                        minI = origin;
                        maxI = origin+stepx+stepy+stepz;
                . .
                        p4 = origin+stepz;
                        p5 = origin+stepz+stepx;
                        p1 = origin+stepx                                    10
                        p0 = origin;
                        p6 = origin+stepy+stepz;
                        p7 = origin+stepx+stepy+stepz;
                        p3 = origin+stepx+stepy;
                        p2 = origin+stepy;
```

Figure 8.9: Extraction Mesh Generator

In order to stress test the speed improvements, three levels of mesh densities are used: a coarse mesh with 70176 cells and 76615 points, a medium mesh with 402144 cells and 422400 points, and a fine mesh with 3328308 cells and 3410064 points.

The measurements have been performed on a Q6600 2.4 GHz machine, with standard system calls. Fig. 8.11 shows the results of the unmodified, naive algorithm, against two level of improvements: the point-in-cell test combined with and without the point in bounding-box test.

It is evident that the naive algorithm is impractical for any large number of mesh extractions. Even for the medium mesh, it takes about an hour to complete. The smarter versions, however, show a considerable improvement in performance – they are better compared in Fig. 8.12.

Fig. 8.12 shows that the two versions have very similar performance for all mesh resolutions. The operations that required the naive algorithm almost an hour on the medium mesh, are now completed in a little more than a minute.

There is a slight decrease in performance on the combined bounding box version. This is because of the additional arithmetic that takes place during each point iteration. On hexahedral meshes, the bounding box test is unnecessary. However, the penalty introduced by the additional code is negligible. On the fine mesh, the difference is less than 5 seconds, and it pays off to insert the additional code whenever the extraction is performed on unstructured meshes, with many higher-order polyhedrons.

Taking advantage of the geometric particularities is worth the cause. The improvements overcome an algorithmic barrier that would have otherwise

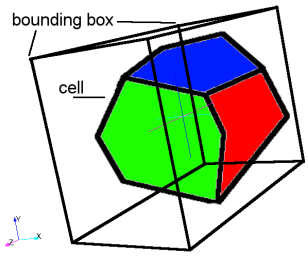


Figure 8.10: Bounding Box over Polyhedron

render the previous solutions impractical, and expensive in real-world applications.

The bounding box implementation is expected to outperform the simple point-in-cell test, on complex, irregular meshes. A point in cell test is basically a point in polyhedron test, where the polyhedron can have an unlimited number of faces. The irregular meshes are required for those regions where user-refinement is necessary, in order to capture the simulated phenomena; they are applied in the difficult portions of the analysis domain.

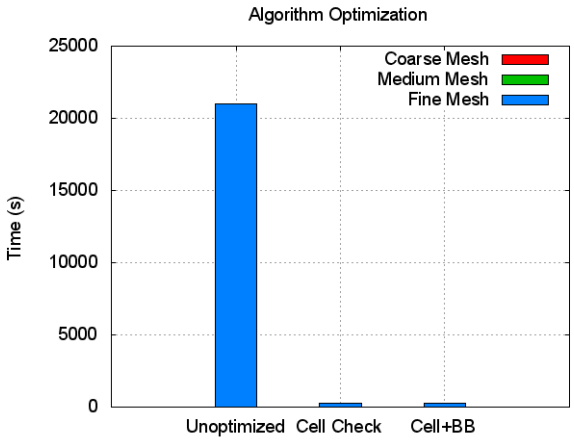


Figure 8.11: Algorithm Comparison

The performance of the extraction algorithm has been improved from several days of running time (depending on the mesh size), to a matter of minutes, by carefully taking into account the geometric particularities of the space-time window. The following sections continue with the two unsteady test-cases, one designed to understand the issue of time interpolation, and one chosen to demonstrate the space-time window concept on real-world, industrial problems.

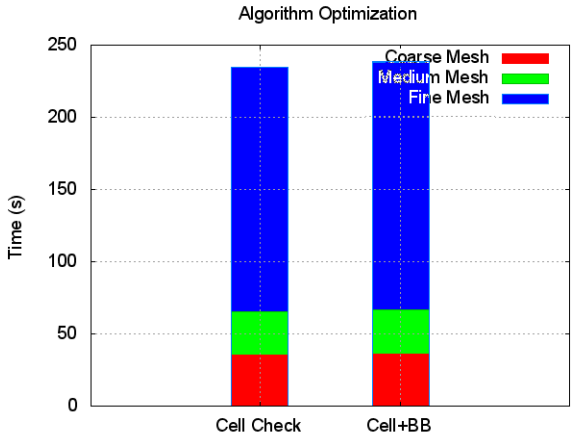


Figure 8.12: Algorithm Performance

8.2 Time interpolation in a specially crafted test case

This test case has been specifically engineered to allow for cut-boundary interpolation through both time and space. That is, we need to make sure that when boundary information for a given time step is missing, the interpolation will not introduce errors. For this purpose, the mesh has been designed so that the flow captured as entering the space-time window, is constant through time, or at least it can be approximated with linear interpolations. The window captures vortices, but only as they form inside the subdomain, travel towards the exterior, and never get back in.

Without this particularity of the simulation, time interpolation introduces inaccuracies at the boundary level, rendering the recalculated flow features incompatible with the global simulation. Since not all the time steps need to be sampled in the space-time archive, because of the time interpolation, the size of the data is considerably reduced.

The domain of analysis in Fig.8.13 is a cavitation problem based on the OpenFOAM tutorial, with a slightly modified mesh [27]. The equations and the solver are described in [133].

The global simulation is a 3D flow using a timestep $\Delta t = 10^{-8}$ s on a mesh with 82238 cells, with a time interval from $t_0 = 3e-06$ to $t_{9996} = 10^{-4}$ s. The space-time window is defined in the region of (7.1, -2, -0.11)(16, 2, 0.11) from $t_0 = 3e-06$ to $t_{1300} = 1.5e-05$ s.

A writing interval of $w_i = 10^{-6}$ s is used, which means that out of 100 time steps, 99 are only used in RAM for computation, and never get to be stored on the disk. In this case, these are the same time steps that get skipped

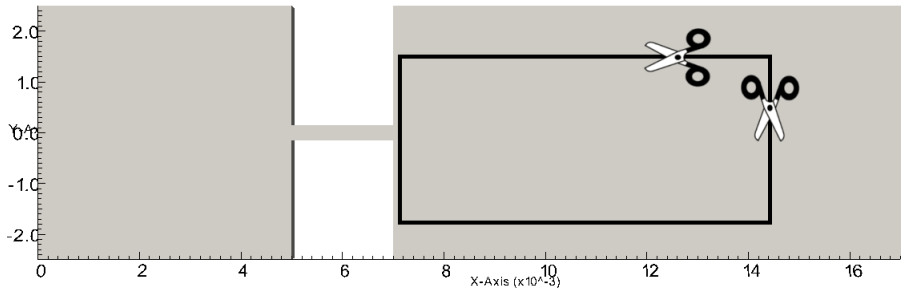


Figure 8.13: Domain of analysis

during the space-time window extraction process, so that when the solution is recalculated, the missing boundary values are interpolated based on the known values from within the encapsulating time interval.

The space-time window uses a $40 \times 30 \times 10$ extraction mesh, with a total of 736800 floating point numbers. This is only $1.93 \times 10^{-5}\%$ from the global simulation, while still preserving the flow features in the space-time region.

Fig.8.14 compares the streamlines for $t_0 = 3e{-}06$ s. Fig.8.14(a) depicts the global streamlines from the initial simulation. In 8.14(b), the reconstructed streamlines are showed for comparison, and they clearly exhibit the same vortex and velocity features as the global version. To better emphasise this, the global streamlines are overlaid with the reconstructed velocity vectors in 8.14(c), so that the same vortex features are more evident. The results in the space-time window are compatible with the global simulation.

The same comparison is performed in Figs.8.15 and 8.16, but on different time steps. The vortices in Figs.8.15(a) and 8.15(b) are shifted to the right, towards the exit of the space time window, and are better captured in Fig.8.15(c) with the velocity vectors.

A streamline is a line in the fluid whose tangent is parallel to the local velocity vector in every point of the flow. Therefore, the family of streamlines at time t are solutions to a given system of equations, and families with various numbers of streamlines can be plotted during the same time step. Thus when comparing the global streamlines against the reconstructed features, it is important to consider that the local space-time window may capture the global streamlines at a different level of zoom, with a different solution space – but always with common main features.

By overlapping streamlines with velocity vectors, it is easy to verify if the rotationary phenomena are correctly captured. The number of velocity

vectors used in the plot is also a matter of visualisation software, therefore the comparison should testify that the main flow features are preserved. The higher level of detail presented in the space-time region is better compared in Figs.8.17-8.18 and Figs.8.19-8.20, where the velocity and pressure profiles directly map the field values.

In Figs.8.16(a) and 8.16(c) a new vortex is captured as it forms nearby the entrance into the space-time region. Fig.8.16(b) does not show this feature because the streamline plotter is using a different level of zoom; however, the main flow features as shown in the central streamlines are preserved. It is also important to notice that the scales for the global and reconstructed velocity fields are almost identical.

The reconstruction of the space time window requires 4912.81 s on commodity, end-user hardware. The global simulation required about one day of work on the same quad-core machine, a Q6600 at 2.4 GHz, using 3 cores out of the 4 available, and covering a time interval until $t = 2.3e - 05$ s.

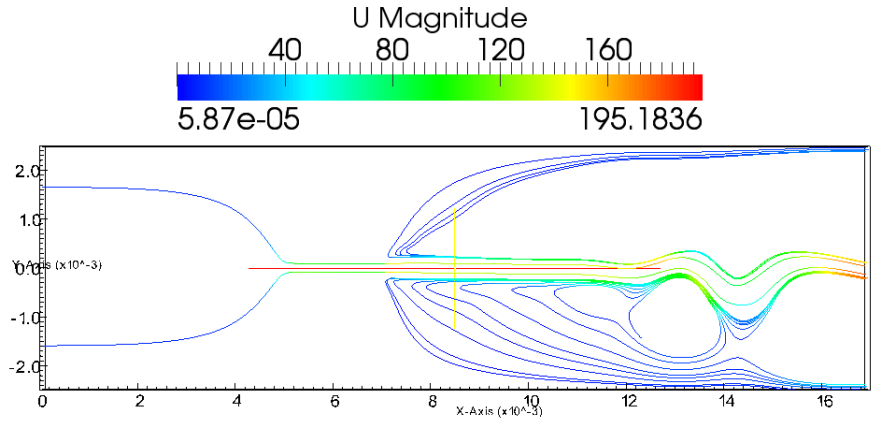
The evolution of the global velocity profiles is given in Fig.8.17, for comparison with the reconstructed data in Fig.8.18. Figs.8.17(a) and 8.18(a), 8.17(b) and 8.18(b), and 8.17(c) with 8.18(c) all confirm that the evolution of the velocity profiles in the recalculated region, during the space-time interval, is compatible with the global simulation.

The pressure velocity profiles evolve as in Fig.8.19, and are compared with the reconstructed version in Fig.8.20. Different snapshots for the same time interval are considered, with 8.19(a) and 8.20(a), 8.19(b) and 8.20(b), and 8.19(c) against 8.20(c). It is clear that the flow features are preserved, and the low-pressure regions which trigger vortex phenomena are shown to shift towards the exit of the space time window.

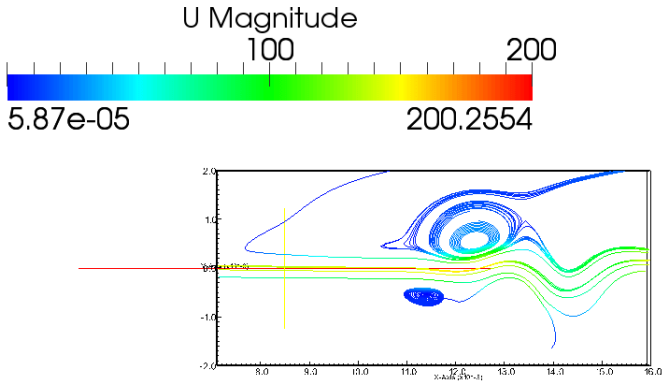
The entire idea behind this test case is to verify when and how time interpolation can be applied. In an unsteady simulation, the boundary patches around the space-time window keep flipping their behaviour towards the analysis subdomain as either inlets, either outlets. This change in behaviour is worsened by any turbulence or nonlinearity present in the cut-boundary region.

The results show that when turbulence can be constrained to patches displaying outlet behaviour, time interpolation becomes possible, due to the fact that the boundary inaccuracies introduced by the interpolation algorithm at the inlet patches, are negligible. This has to hold true for each of the simulation time steps.

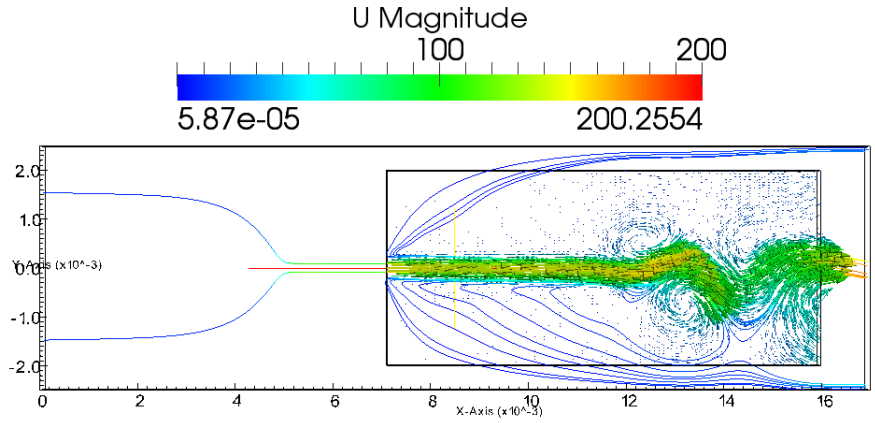
The problem of identifying natural phenomena where these constraints can be satisfied is difficult, and it still is a matter of debate. However, it is important to remark that time interpolation strongly amplifies all the benefits of the space-time window reconstruction concept.



(a) Global streamlines

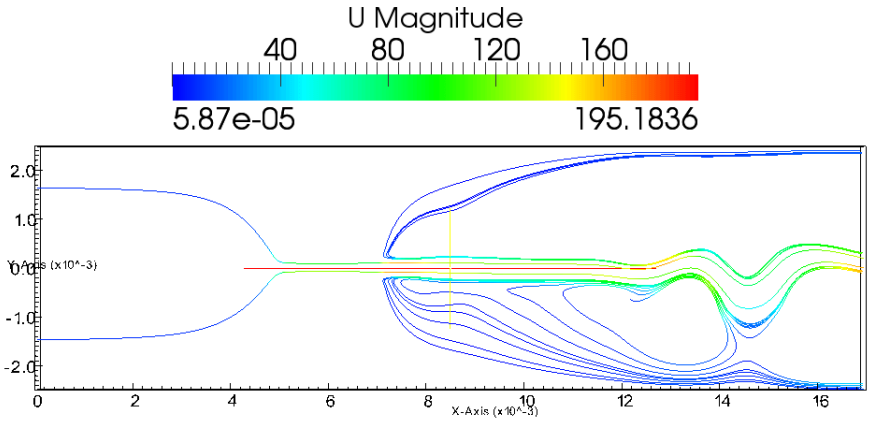


(b) Reconstructed streamlines

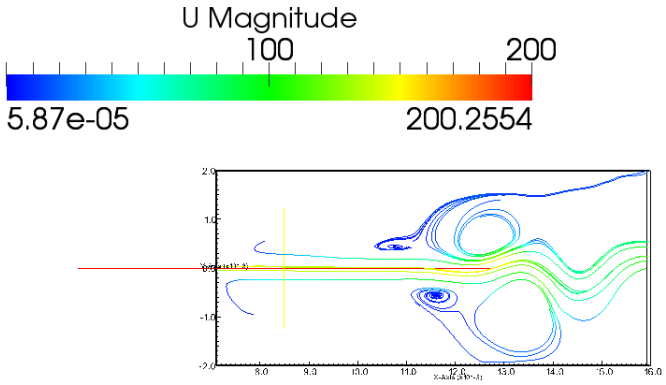


(c) Reconstructed velocity vectors against global streamlines

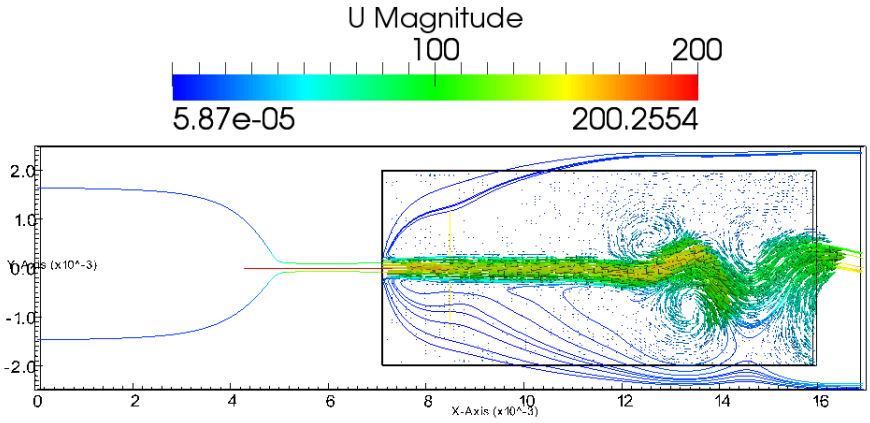
Figure 8.14: Streamlines and velocity vectors at $t = 3e - 06$



(a) Global streamlines

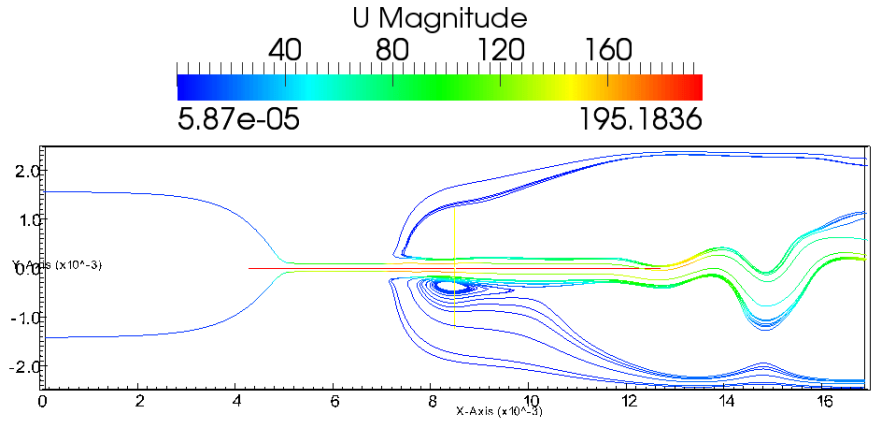


(b) Reconstructed streamlines

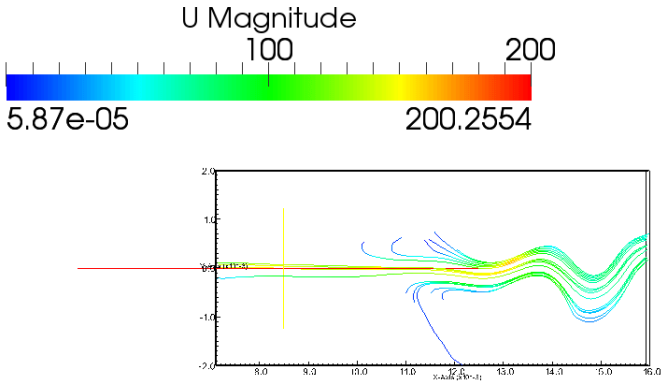


(c) Reconstructed velocity vectors against global streamlines

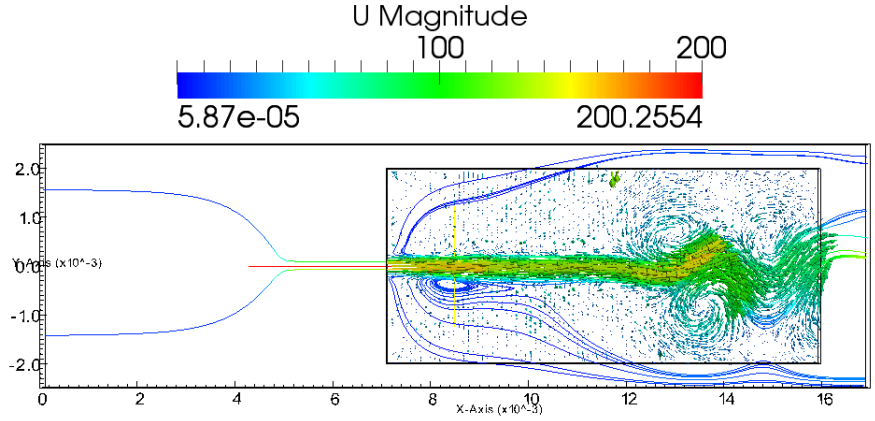
Figure 8.15: Streamlines and velocity vectors at $t = 9e - 06$



(a) Global streamlines

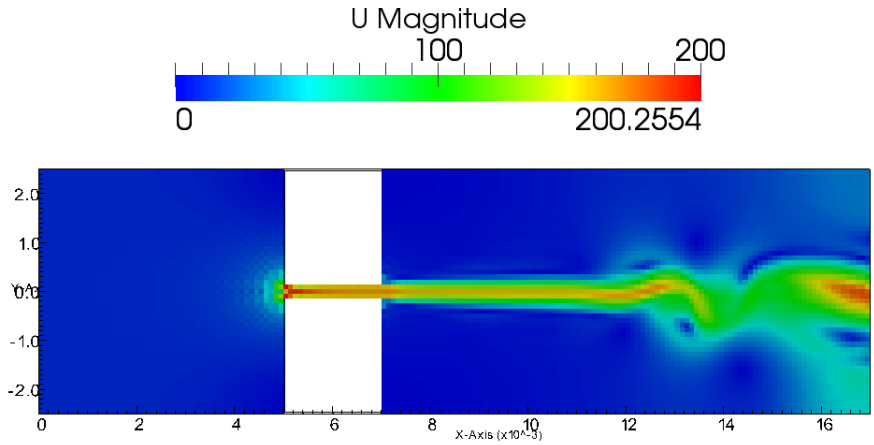


(b) Reconstructed streamlines

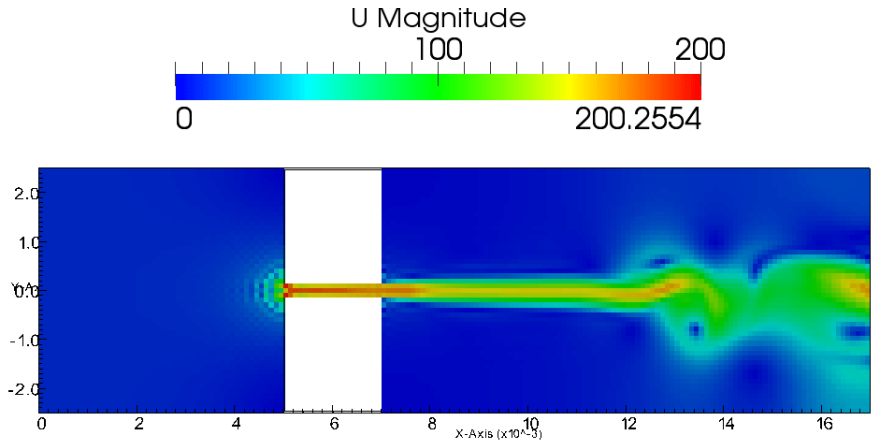


(c) Reconstructed velocity vectors against global streamlines

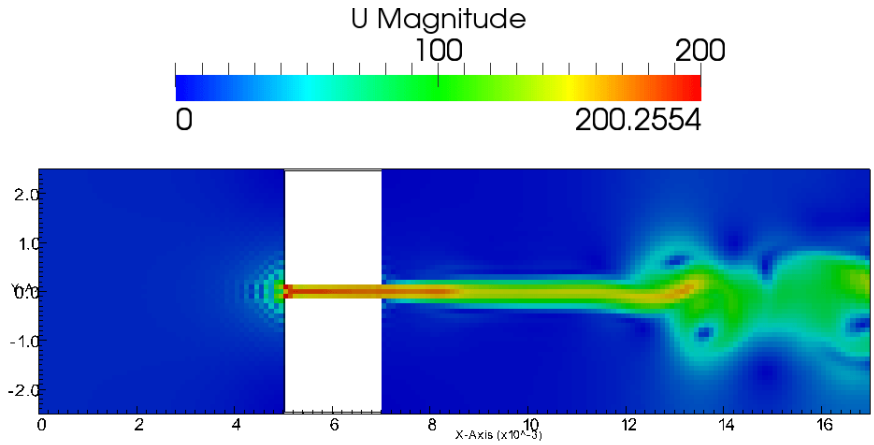
Figure 8.16: Streamlines and velocity vectors at $t = 1.5e - 05$



(a) $t = 3e - 06$

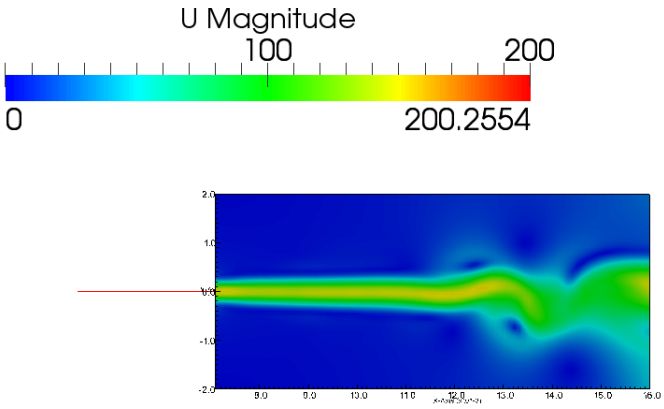


(b) $t = 9e - 06$

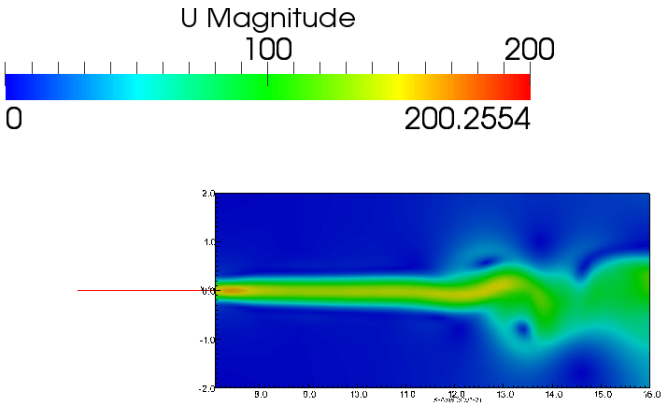


(c) $t = 1.5e - 05$

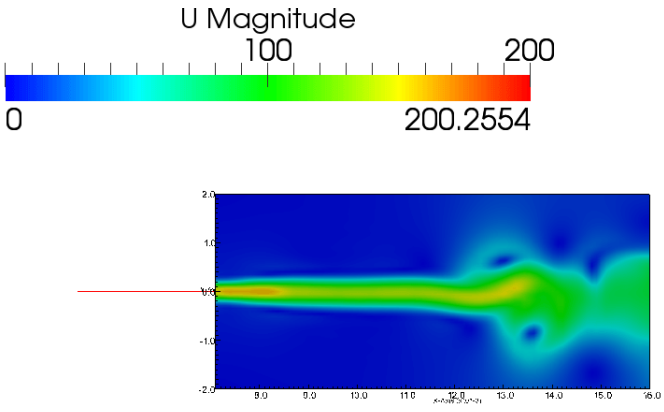
Figure 8.17: Global velocity contours



(a) $t = 3e - 06$

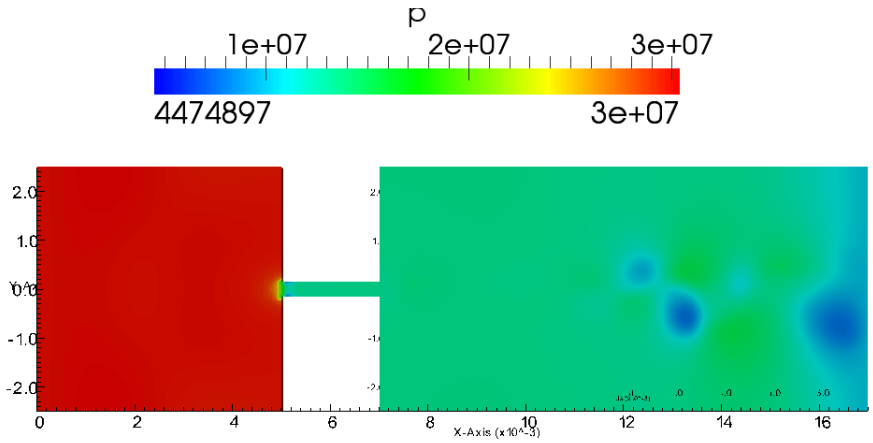


(b) $t = 9e - 06$

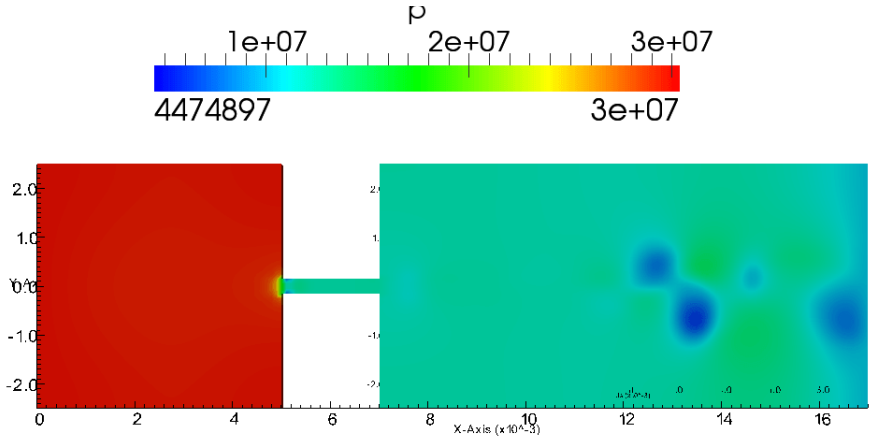


(c) $t = 1.5e - 05$

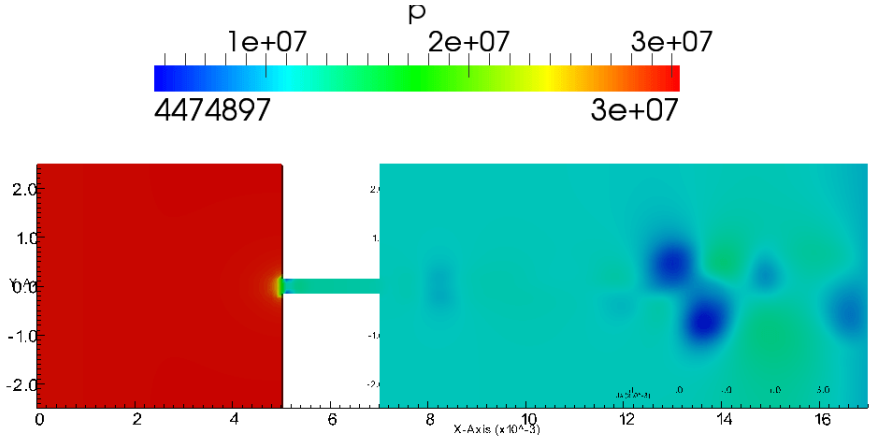
Figure 8.18: Recalculated velocity contours



(a) $t = 3e - 06$

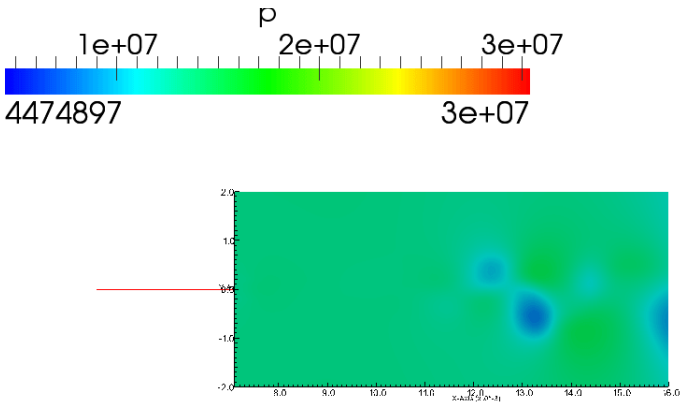


(b) $t = 9e - 06$

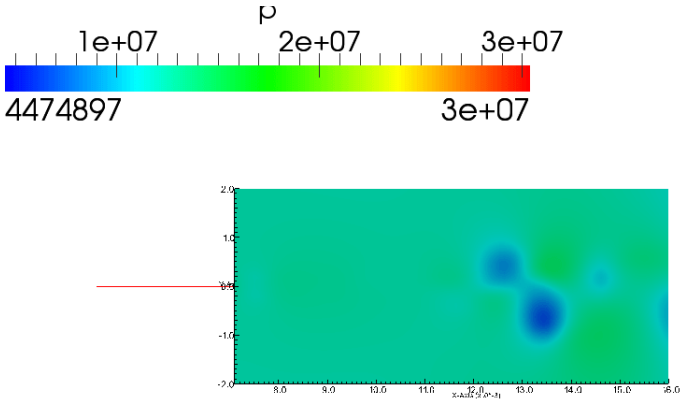


(c) $t = 1.5e - 05$

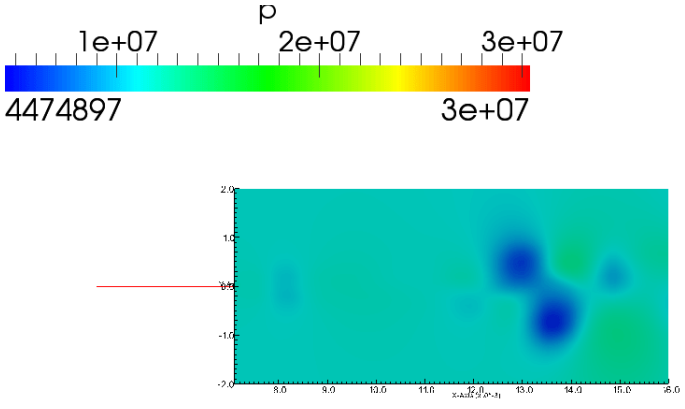
Figure 8.19: Global pressure contours



(a) $t = 3e - 06$



(b) $t = 9e - 06$



(c) $t = 1.5e - 05$

Figure 8.20: Reconstructed pressure contours

The equations used throughout this thesis are infinitesimal, describing continuum mechanics. Because of that, it is always possible to zoom-in further in the simulation, and capture more details. It is up to the physics to describe a phenomenon where all the flipping inlet boundary conditions can be approximated as linear or even constant, through time.

For now, time interpolation has been shown to be possible, given that the necessary constraints are satisfied, but only in a test case where the mesh has been engineered to support it. A refined mesh for the very same test case starts capturing turbulent phenomena at the main space-time window inlet, rendering all time interpolation efforts unsuccessful.

The next section uses a real-world simulation, which has been validated against experimental data and against results from literature. Time interpolation is not used any more, since the boundaries are highly nonlinear, and the cut region is located in the middle of a turbulent flow. For the space-time window reconstruction concept to be applied, information from all of the time iterations has to be extracted and stored. The thesis will now continue with the most complex scenario, for the implementation based on submodelling.

8.3 The ERCOFTAC square cylinder benchmark

The square cylinder benchmark is a well known ERCOFTAC [120] test-case. The setup is an OpenFOAM [27] implementation.

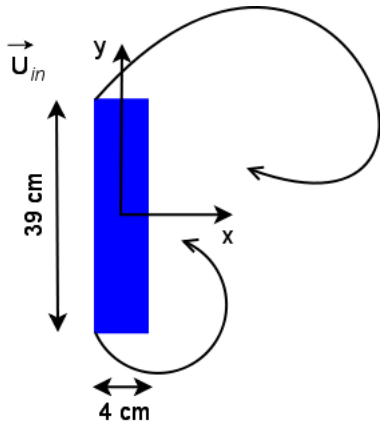


Figure 8.21: ERCOFTAC Square Cylinder

Consider the square cylinder presented in Fig.8.21. The inlet velocity is $U = 0.535\frac{m}{s}$ and the Reynolds number is $Re = 21400$. More details are given in Lyn and Rodi [120].

$$\nabla \cdot (\rho \vec{v}) = 0 \tag{8.5}$$

$$\frac{\partial(\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v}\vec{v}) = -\nabla p + \nabla \cdot \vec{\vec{\tau}}' + \rho \vec{f}_m \tag{8.6}$$

An LES simulation is performed in order to show the Karman vortex shedding that takes place behind the cylinder as in Fig.8.22.

The governing equations are given by continuity and momentum as described in (8.5) and (8.6), where $\vec{\vec{\tau}}'$ is the viscous stress tensor and f_m are body forces.

The centre of the cylinder is configured as a reference frame. The diameter of the square cylinder is $D = 0.04$m. The overall bounding box of the analysis domain, in meters, is given by (-0.4 -0.28 0) (0.96 0.28 0.392).

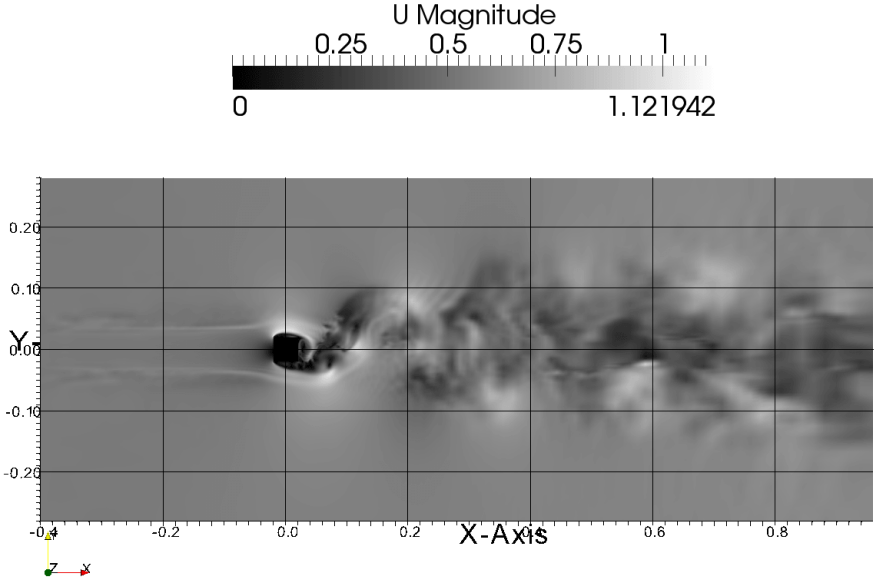


Figure 8.22: Karman vortex street

Karman vortex streets are frequently found in nature. High-altitude mountain peaks, islands, skyscrapers and cables are known to produce vortex streets.

Perhaps the most frequent encounter is provided by industrial chimneys. When the wind produces street vortices behind the chimneys, it induces structural stress forces that can be dangerous. Spiral ridges are used to attenuate the phenomenon.

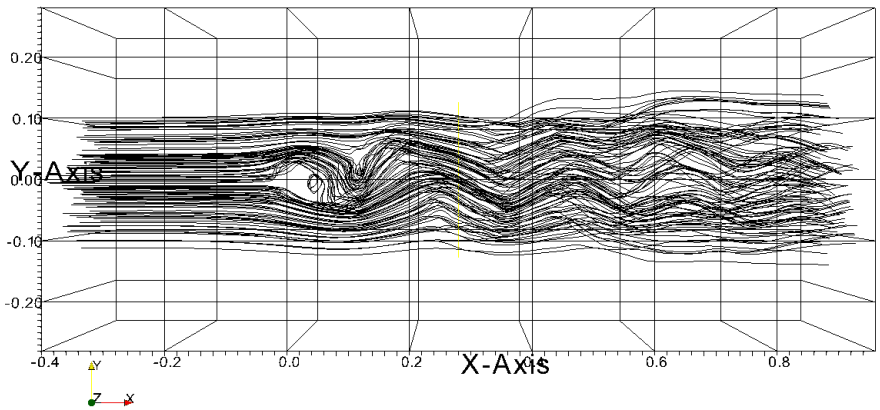


Figure 8.23: Streamlines for the Karman vortex street

Fig.8.23 depicts the streamlines that form the vortices behind the square cylinder obstacle. Fig.8.24 shows the LES simulation result at 5 seconds distance from the initial start-up.

In Fig.8.25 a space-time window of 75 time steps, described by a bounding box with coordinates (0.045 -0.1 0.05) and (0.15 0.1 0.35), is extracted starting with $t = 5$s. The sampling mesh is defined by $99 \times 99 \times 99$ cells, with 100 sampling points for each dimension. The global LES simulation has more than 3.3 million cells.

In order to reconstruct the internal fields, a new mesh, of $100 \times 100 \times 100$ cells, is generated. The boundary values from $t_1 = 5$s up to $t_{75} = 5.0075s$ are linearly interpolated on the new mesh, using the extracted values. With a $\Delta t = 10^{-4}$s, the boundary conditions for 75 time iterations are stored into the space-time window capsule.

The flux Q is defined in (8.7), where $\vec{U}$ is the velocity and $\vec{n}$ the normal to the surface dS.

$$Q = \int_S \vec{U} \cdot \vec{n} dS \tag{8.7}$$

For incompressible fluids (with constant ρ density), the conservation laws impose that in Fig.8.26 $Q_{inflow} = Q_{outflow}$, at any moment in time. This must hold true in order to satisfy (8.5), basically stating that the mass getting into the space-time window equals the mass exiting the space-time window, for

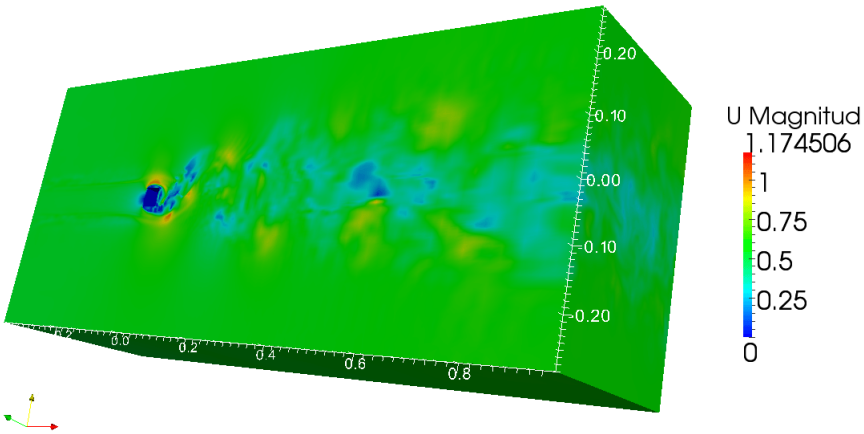


Figure 8.24: Global Domain Simulation

each of the 75 time steps; the mass conservation is said to be preserved.

However, the extraction of the space-time window is based on linear interpolation. The global simulation is based on nonlinear systems of PDEs. This results in conservation errors that need to be monitored and controlled.

For the purpose of this research, the acceptable error range is considered to be less than 1%. This is computed using the formula in (8.8) for each time step in the space-time window.

$$C = \frac{|Q_{inflow} - Q_{outflow}|}{Q_{inflow}} \times 100 \tag{8.8}$$

Fig.8.27 depicts the global streamlines passing through the window subdomain, and the extraction edges. The mass conservation inside the space-time window is preserved in acceptable ranges. For time step t_1, the mass conservation error is $C_1 = 2.5 \times 10^{-1}\%$. It slowly increases until the last time step, at t_{75} being $C_{75} = 1.13 \times 10^{-1}\%$. The magnitude of the conservation error depends on the nonlinearity of the boundary phenomenon, and it may increase or decrease through time.

In OpenFOAM, the functionality required to configure the space-time window for reconstruction is provided by time-varying boundary conditions. The recalculated window subdomain is depicted in Fig.8.28. The submodelled streamlines clearly follow the path found by the global simulation, as can be seen in Fig.8.29.

The velocity contours from the submodelled window are also in agreement with the streamlines from the global simulation.

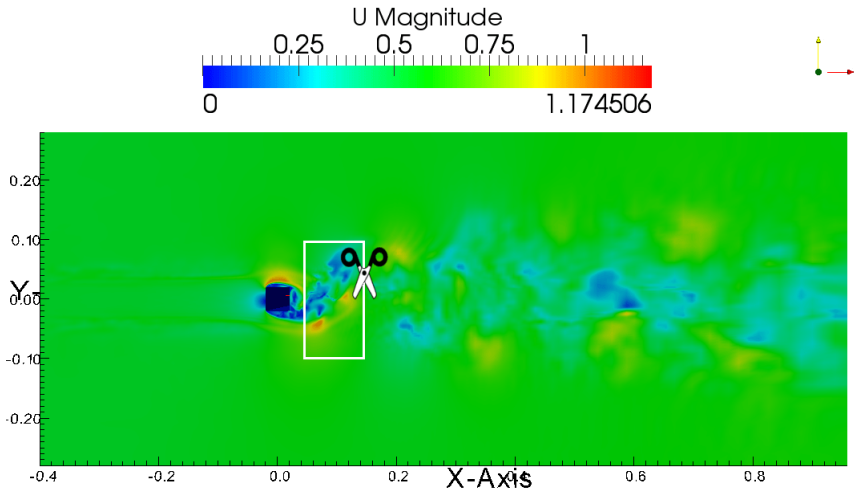


Figure 8.25: Extraction of The Bounding-Box Behind the Square Cylinder

In fact, in Fig.8.30, after 75 time iterations, the submodelled window captures half of a vortex as it enters through the surface.

The complete simulation would require 15 Gb of data to be downloaded and archived, after being subsampled to 0.5 s intervals. Such a subsampling prohibits the visualisation of the result in real-time animations. More realistic subsamplings vary from $100Gb$ to $1Tb$, depending on the requirements. LES always produces large results, because it requires very fine meshes.

The global simulation, generated on 8 distributed processors takes more than 24h for less than 3 seconds of simulation. For 20 seconds of simulation, a 256 processors cluster is used for 24 hours of work, thanks to the bwGRiD supercomputers [1]. On the other hand, reconstructing the space-time window on end-user hardware takes about 1 minute per time iteration. This allows for high-resolution numerical results to be obtained again on commodity

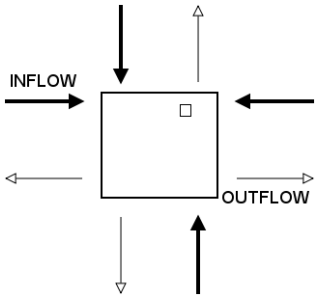


Figure 8.26: Inflow and Outflow through the Space-Time Window

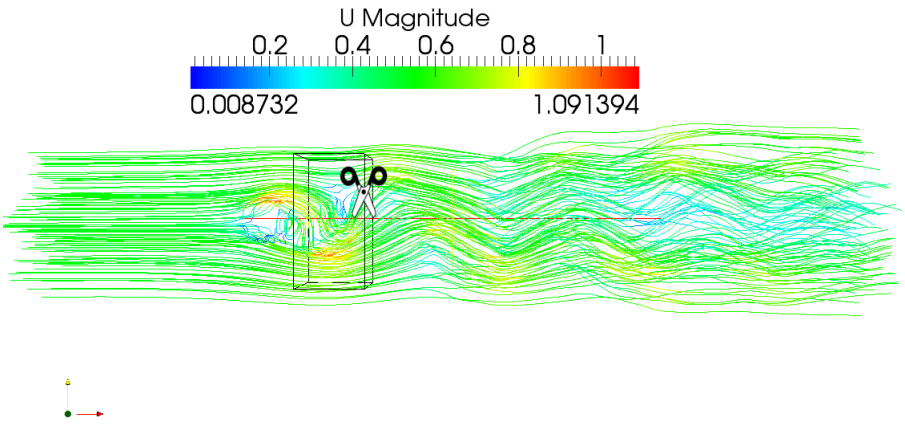


Figure 8.27: Global Streamlines through the Window Subdomain

hardware, without any use of expensive computers.

The mesh in the space-time window is 5.43 times finer than the global mesh in that region. This produces a zoom-in effect, allowing for better details to be captured in the space-time window. Even so, the flow features are clearly preserved between the reconstructed and the global version of the simulation.

Fig.8.31 shows the pressure isocontours that form in the initial simulation. Figs.8.31(a), 8.31(b) and 8.31(c) are snapshots taken after a different number of time iterations.

In order to better follow the phenomenon, Fig.8.32 shows the Karman vortex street 'in motion', using 3D pressure contour features. The 3D vortices from Figs.8.32(a), 8.32(b) and 8.32(c) correspond to the profiles in 8.31(a), 8.31(b) and 8.31(c).

Using the space-time data, the vortex features inside the window are independently recalculated, with the same solver, as shown in Fig.8.33. The 3D vortices in Figs.8.33(a), 8.33(b) and 8.33(c) are obtained after a different number of internal solver iterations for each of the 75 time steps. In other words, 75 time iterations in reality produce a total number of numerical iterations which is an order of magnitude higher, due to convergence loops.

Unlike the previous test-case, where time interpolation has been shown to be possible, in this case, time interpolation is no longer possible. This means that data from all of the time steps needs to be extracted and stored for the space-time window. In reality, not all these time steps are stored to the disk, and it is a common practice to subsample the global simulation.

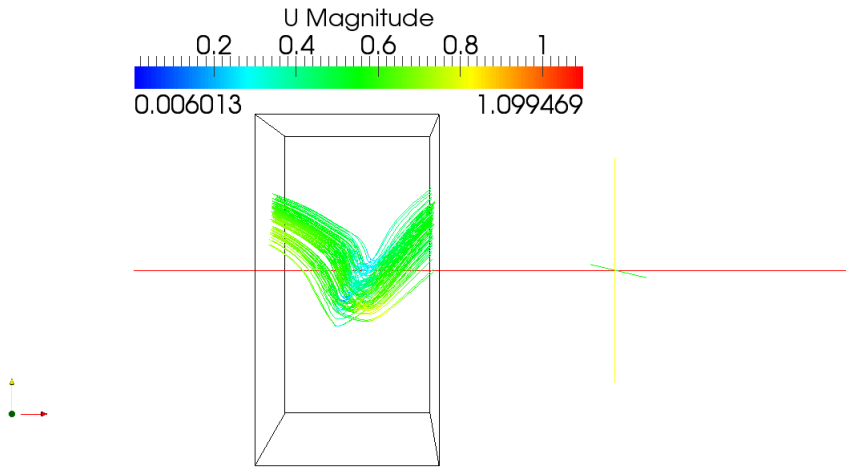


Figure 8.28: Submodelled Streamlines

However, space-time information needs to be extracted from all the time steps, including those which do not normally get written to the disk. The problem can be alleviated through implementation improvements, but nevertheless, the size of the space-time window data is increased.

On the other hand, the smaller the time step Δt is, and the larger the time interval covered by the space-time window has to be, the bigger the size of the data in the space-time window becomes.

Currently, there are two levels of interpolation present. The first one is performed when the data is sampled from the global simulation. The second one is used when the extracted data has to be mapped on the new, reconstruction mesh. Instead of using two levels of interpolation, the implementation could replace the first one by extracting only calculated data from the global simulation, so that the bounding box of the space-time window is contained by the coordinates of the sampled points. This also removes the problems with the resolution of the extraction.

For now, data is sampled according to a regular hexahedral mesh. This is suboptimal in terms of the number of necessary coordinates. For instance, regions in space and time where data is coarse do not require finer resolutions for extraction, but data is extracted at regular distances anyway. Also, in regions where the calculated information is dense, the extraction mesh can be too coarse and miss important bits, unless the entire space-time extraction has enough resolution. The higher the extraction resolution, the more data needs to be stored in the space time window, with or without practical utility. Even so, the results are remarkable.

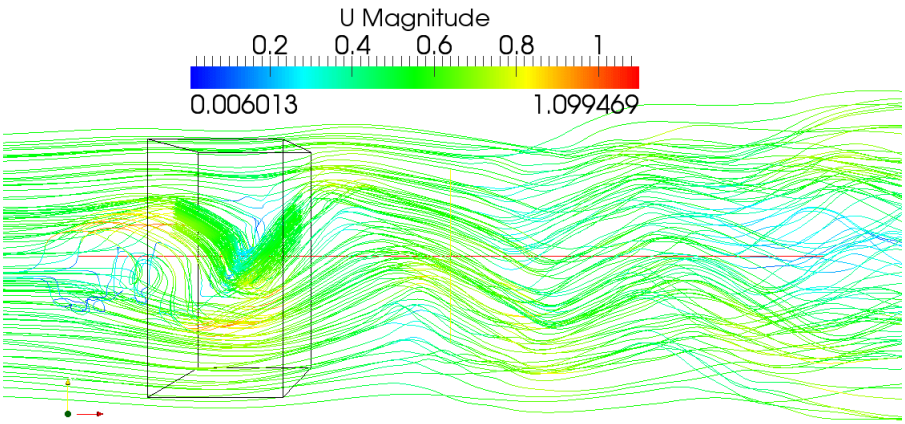


Figure 8.29: Submodelled Streamlines follow Global Streamlines

The interpolation algorithm is linear in both 2D and 3D space. Tetrahedral interpolation is used for the initial 3D fields, and it works well on unstructured meshes with higher-order polyhedrons; however, the results may be different depending on the way a particular mesh is formed and decomposed into tetrahedra. The current implementation uses one form of decomposition of regular hexahedra into 6 tetrahedra, and this suffices for the state of the development.

Modern supercomputing faces three levels of bottlenecks: HPC facility level, gateway level, and user level. The first level can be alleviated by using the space-time window extraction tool remotely, inside the supercomputer facility. This allows the user to archive, compress and download only the space-time window information from within the HPC centre. Due to the fact that supercomputers usually benefit from high performance, networked filesystem services, the reconstruction phase can not be executed on HPC systems, with the present implementation. The current implementation makes use of a considerable amount of small files, required by the OpenFOAM classes, a particularity which may create serious problems when run on such filesystems.

The problem can be solved by modifying the implementation, however, the reconstruction is not designed to be run on supercomputers, but on end-user hardware.

For solvers which require very small time steps, like $\Delta t = 10^{-8}$ s, the reconstruction process can be further accelerated on the user machine by taking advantage of the affordable technologies. General-Purpose computation on the Graphics Processing Unit (GPGPU), or many-core machines can

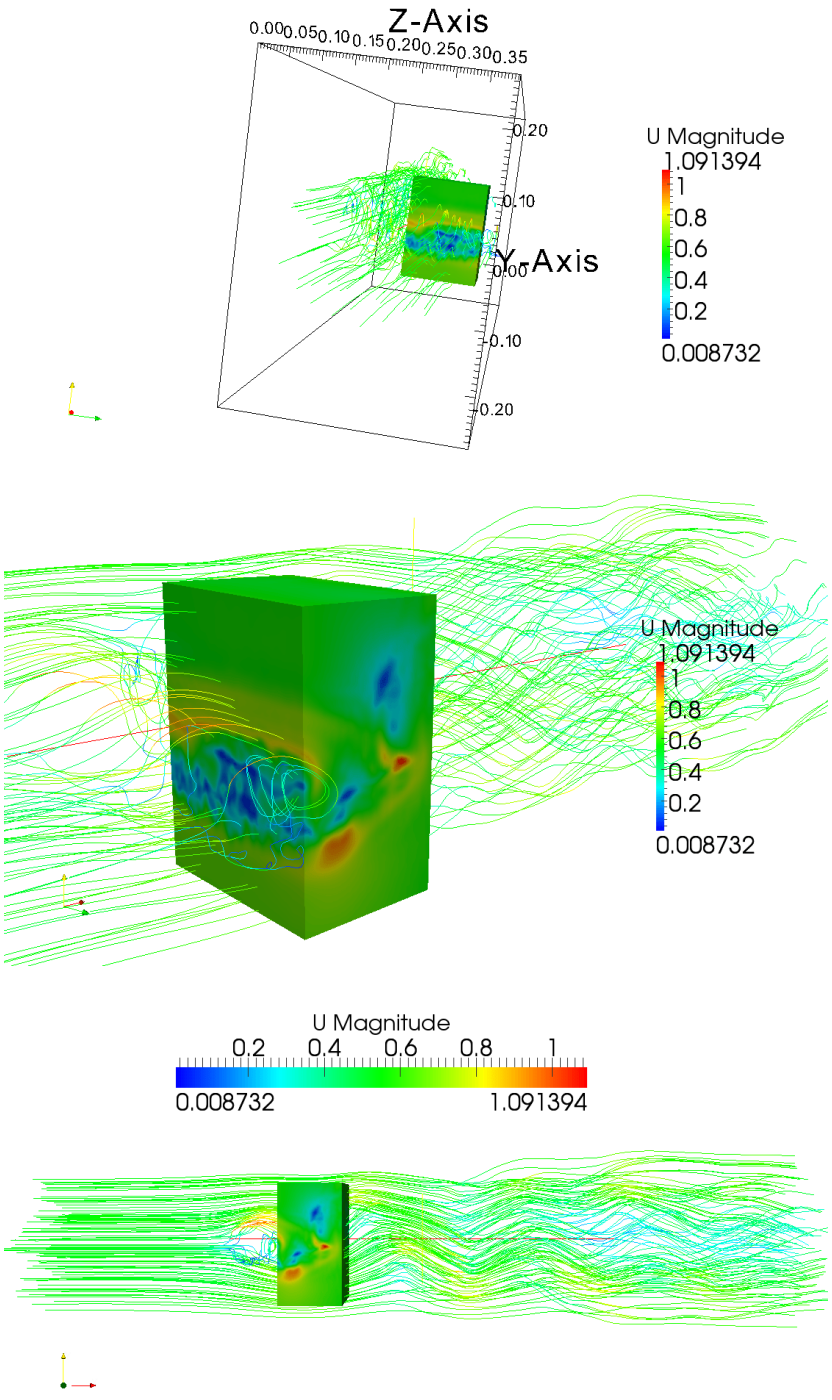


Figure 8.30: Global Streamlines vs Submodelled Velocity Contours

be used to massively parallelise the matrix-vector operations. FPGAs can be used to speed-up the computation. Therefore, the space-time window reconstruction concept has great potential for commodity hardware.

The second and the third level of bottlenecks are implicitly relieved from the data deluge, because the data that needs to be transported throughout the network and the Internet is reduced to one or more space-time windows.

In Fig.8.34, the reconstructed pressure isosurfaces are compared with the global stream lines. It is clear that the reconstructed fields inside the space-time window retain the same flow features from the global simulation.

After all time steps converge, the vortex features do not present any sign of numerical diffusion. This means that the method is feasible.

The space-time window for the complete set of 75 time steps requires about the same number of floating point entries as a single time step in the global simulation, which is 46600000 64-bit numbers. This makes it occupy much less space than a complete simulation where, like in this case, 100 global time steps are written to the disk, and the data is practically reduced to 0.87%.

The possibilities and the flexibility that are introduced, alongside with the alleviation of the bottleneck problems, are virtually limitless. Not only that the data deluge problem is solved, but the user is given back his freedom over the analysis.

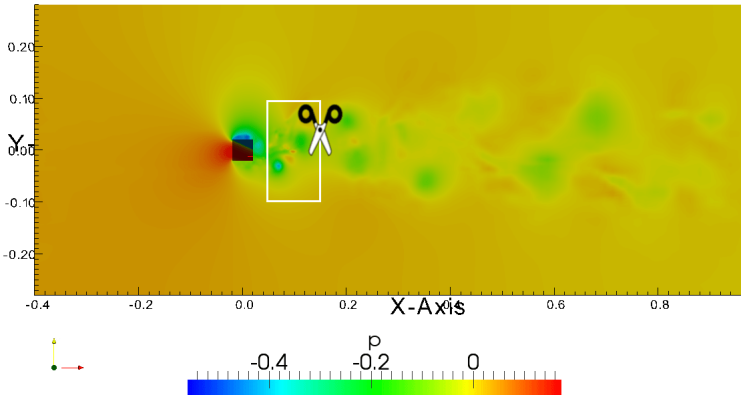
Instead on depending on the global simulation, the user can improve the mesh, and even change to more appropriate dynamic models inside the space-time window. For instance, considering that this test case presents the flow over an obstacle, additional obstacles can be introduced, gradually or not, in the space-time reconstruction region.

On the other hand, the user can now calibrate the results for or against other independent simulations, space-time windows or not. For example, the refined and reconstructed space-time region can be used as input for a new simulation. The implications in multi-scale analysis and multiphysics are major.

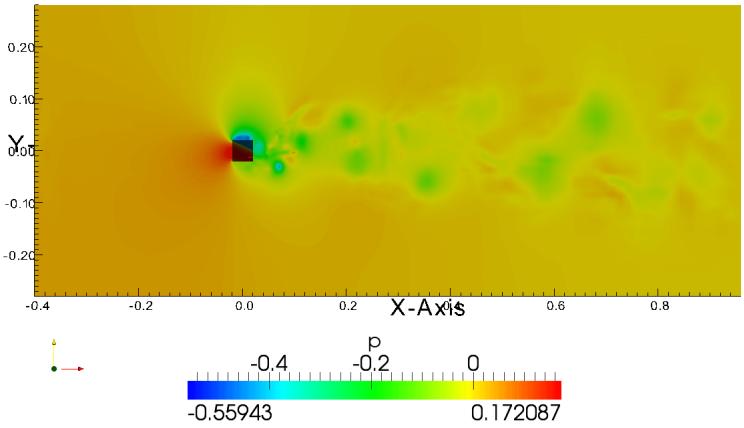
For a more appropriate comparison between the global and the reconstructed fields inside the space-time window, Fig. 8.35 plots the evolution of the scaled percent difference through time. The scaled percent difference is defined by (8.9), where e_g and e_r are the global and reconstructed fields, with the scale given by the e_{max} and the e_{min} values at a given time step.

$$D = \frac{|e_g - e_r|}{|e_{max} - e_{min}|} \tag{8.9}$$

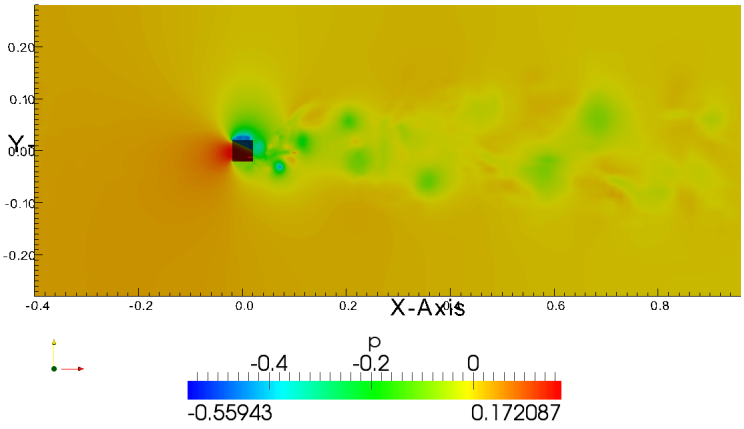
The author considers a vector to be more prone to modifications through time. For this reason, the velocity vector field U has been chosen as a basis



(a) $t_{idx} = 0$

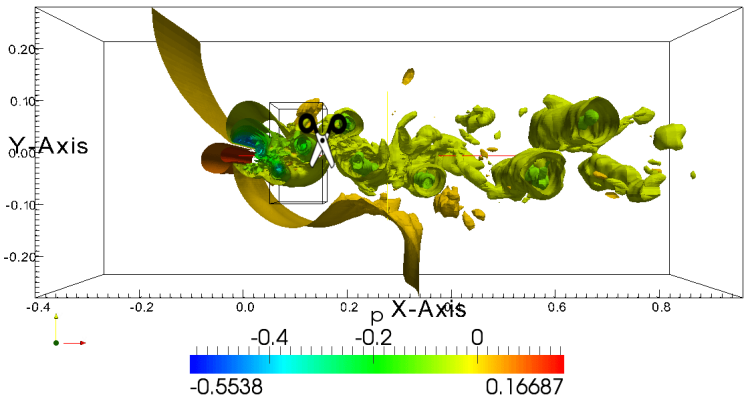


(b) $t_{idx} = 35$

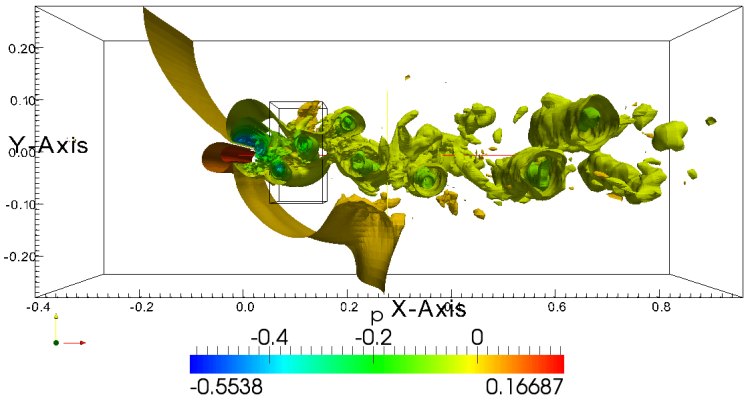


(c) $t_{idx} = 75$

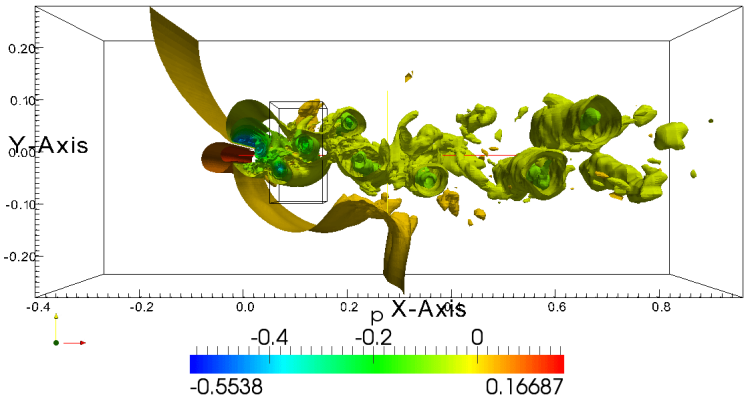
Figure 8.31: Global pressure fields



(a) $t_{idx} = 0$

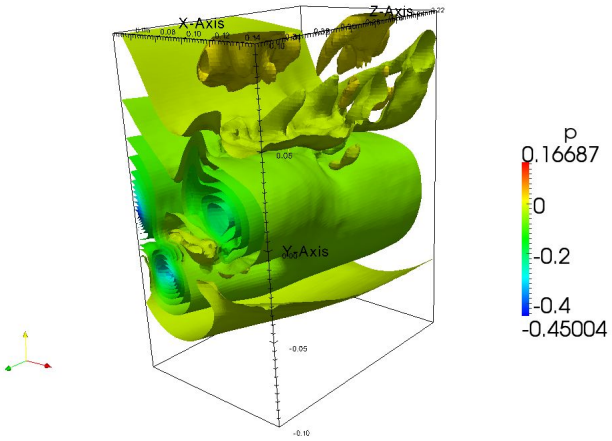


(b) $t_{idx} = 35$

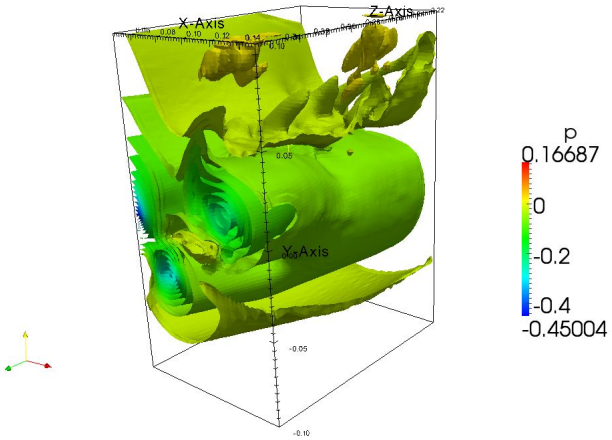


(c) $t_{idx} = 75$

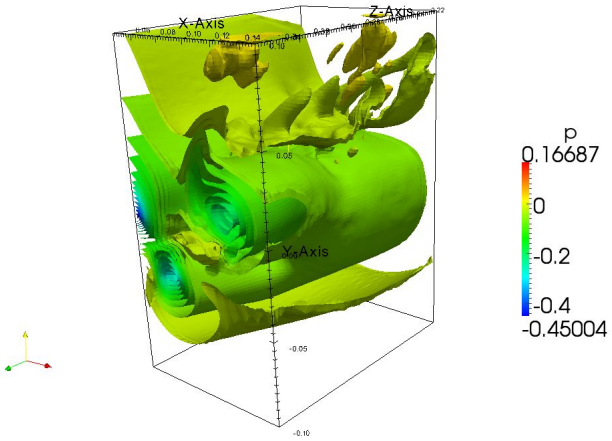
Figure 8.32: Vortex street in motion



(a) $t_{idx} = 0$



(b) $t_{idx} = 35$



(c) $t_{idx} = 75$

Figure 8.33: Reconstruction of the vortex features inside the space-time window

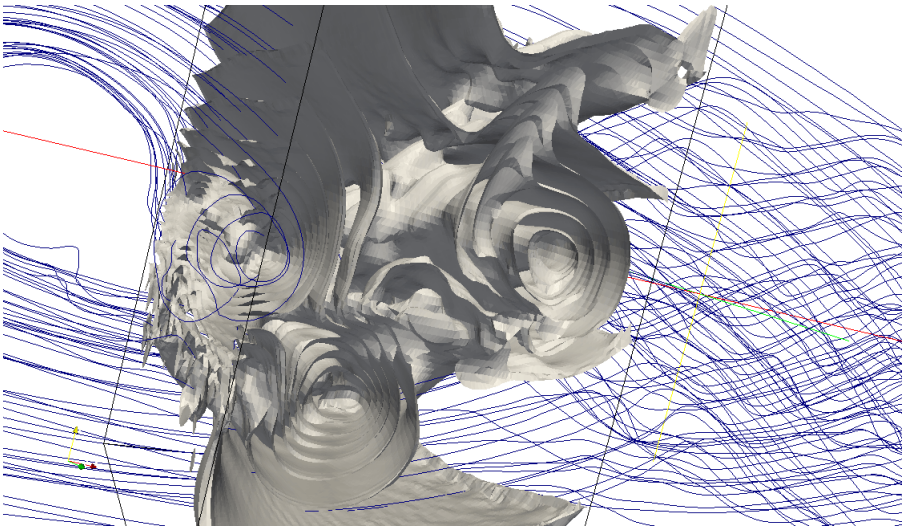


Figure 8.34: Submodelled Streamlines vs Reconstructed Pressure Contours

for comparison between the global simulation, and the space-time window fields. The U_x component is parallel with the direction of the flow, and is used in the comparison as a representative for the U vector, due to the fact that it has the highest potential to change between time steps.

Fig.8.35 displays both the maximal and averaged values for the scaled percent difference D. The D_{avg} increases very slowly from nearby 0 to $D_{avg} \leq 0.5\%$. This definitely proves the point that the space-time window reconstruction results are in agreement with the global simulation, in spite of possible zoom-in effects.

In fact, the slow increase in the average difference D_{avg}, reflects the cumulative nature of the interpolation approximations and the higher level of mesh refinement used in the space-time window reconstruction.

The maximal difference D_{max} varies between $9 - -21\%$. These are high values, but are explained in Fig.8.36.

As shown in Fig.8.36, the number of points that have a scaled percent difference higher than 5% is below 0.32%, from the total of 10^6 centres in the reconstruction mesh. In other words, the reconstructed space-time window fields are in solid agreement with the global simulation.

The evolution throughout the 75 time steps shows a relatively constant increase, with a speed-up after 40 time iterations. This can be explained by the space-time window capturing the internal fields at a higher resolution level.

All things considered, the space-time window reconstruction based on submodelling can not only be used to obtain the same flow features as in the

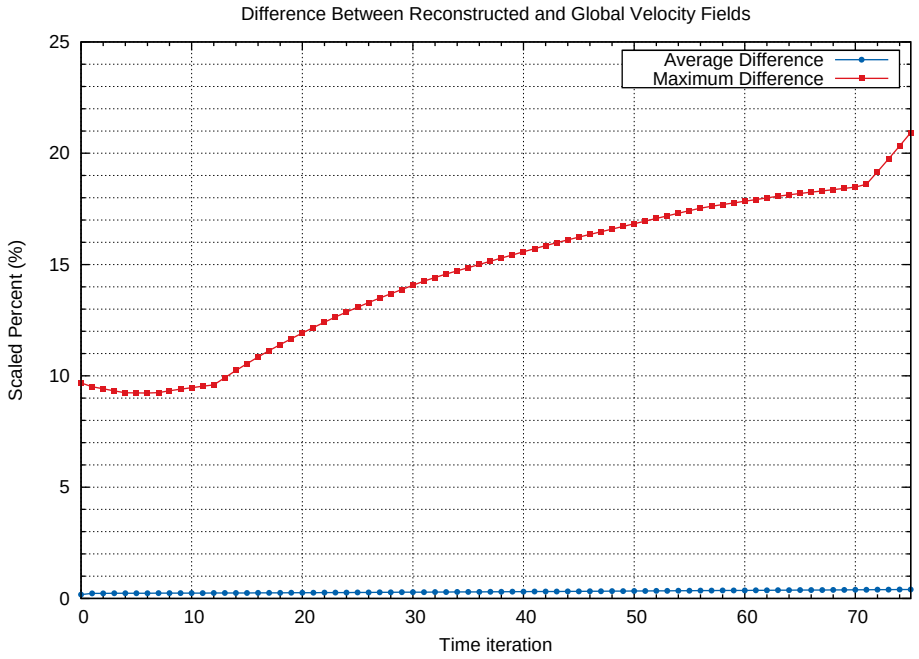


Figure 8.35: Scaled percent difference between reconstructed and global velocity

global simulation, but it can also be used to reconstruct the internal fields with acceptable accuracy.

However, the main goal of this solution is not the reconstruction of the very same fields, but the flexibility and freedom given to the user, to handle the flow features outside the global context, unhindered by the technology.

The direct comparison of the fields, between the space-time window and the global simulation, may provide a quantitative method for validating the reconstruction, but it is by far not the best choice. Reconstructing the fields in continuum mechanics may easily introduce phase shifts in both time and space. These can not be dealt with solely in a quantitative manner. The space-time window is basically a new, independent simulation, based on the global version, and requires hydraulic revalidation, just like the global simulation does.

Laser measurements can provide a reference frame, for both the global simulation and the space-time window, in order to better control and monitor the possible spatial and temporal shifts. However such and approach is beyond the purpose of a computer science thesis. Therefore, it can be concluded that the space-time window reconstruction based on submodelling is a robust solution for alleviating the data deluge problem, but has powerful

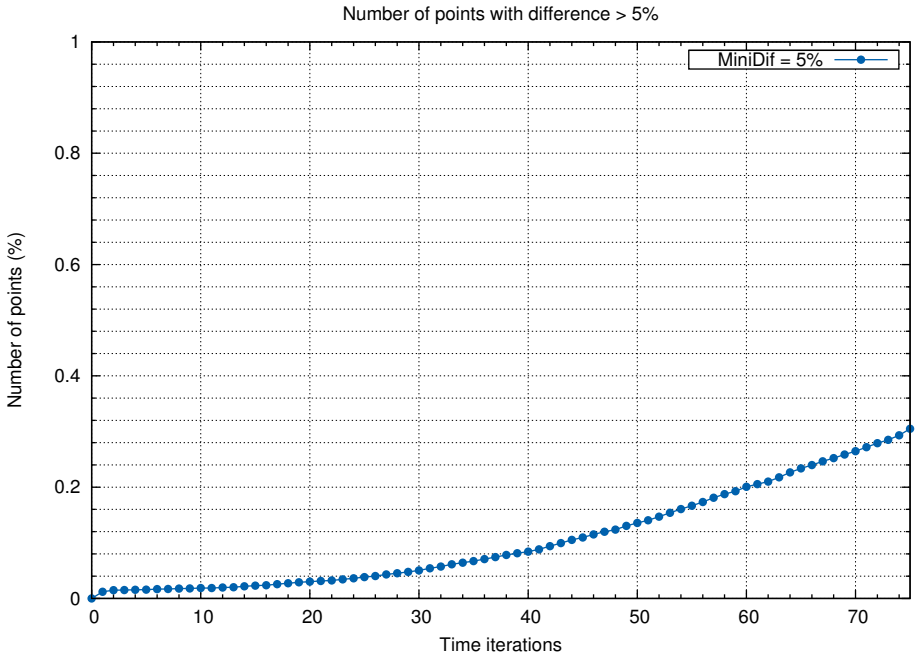


Figure 8.36: Number of points with difference ≥ 5 %

and promising features for empowering the CFD user with freedom and flexibility.

This chapter has been designed to push the approach based on submodelling to practical complexity levels. Even though the tool that has been developed plays the role of a software 'booth', the results are promising and founded on solid ground. The next section continues by drawing the main conclusions around the implementation, and the two test-cases that were chosen.

8.4 Concluding remarks

Not all supercomputing applications are simulations, and not all simulations are numerical simulations. However, most of the applications in CSE involve numerical simulations, and a large part of the global computational resources is allocated to the CFD community.

As already mentioned, the data reduction is just a side effect for the submodelling approach. The strength of the space-time window reconstruction based on submodelling relies in the freedom and flexibility given to the user. The user is now able to create independent high-resolution simulations on

cheap commodity hardware, based on large-scale supercomputer versions. Several degrees of freedom are given foundations: the abilities to independently change the dynamics model, change the mesh, add interaction with other test-cases, and to use microscopic effects at different levels.

The author has learned, in a very tedious way, that it is possible to reconstruct the fields and the flow features in the middle of a turbulent, real-world, unsteady flow, with the help of submodelling. This is a leap forward for the state of the art, and lays down new foundations for the CFD community, spreading the seeds for a shift in the way many numerical simulations are performed.

CFD numerical simulations have complex methodologies for receiving validations against experimental data. First, global flow coefficients must be computed – for instance the Strouhal number, together with drag and lift coefficients. After these are confirmed, a pressure signal can be used as a reference for triggering laser velocity measurements. However, all this is by far hydraulic engineering, and requires a certain level of expertise and background in the field, in order to be performed; it is out of the scope of a computer science thesis. The software tool that has been developed for this solution serves as a 'booth', showing solid results, and requiring contributions from experts in hydraulics and applied mathematics.

Many aspects of the implementation can be improved. Besides that, several modifications can make the life of an investigating engineer easier – such as the reduction of the number of levels of interpolation to only one, which is still necessary during the reconstruction phase.

Another modification is related to the way space-time window data is configured for the reconstruction, and requires the implementation of a new OpenFOAM boundary class. The default one uses a large number of small files with temporal data, and raises memory issues for a larger number of time steps. All this can be alleviated through a new boundary class, capable of reading the space-time archive directly, and reading the necessary values on the fly.

It may be that normalisation of the vector fields, in order to respect the mass conservation ad literam, can help the numerics to swallow the input more easily. If that is the case, a 'self-healing' capability of the new boundary class may come at hand.

For the submodelling approach to be sound, flux conservation should be ensured. In Jasak [78] the author introduces a method for preserving the flux conservation during adaptive mesh refinement. This is achieved by solving the pressure equation again on the new mesh, and deducing the fluxes. Due to the conservative boundary conditions, the obtained fluxes are also bound to be conservative. This is easily achieved in a classic mesh refinement context.

However, submodelling implies that the new simulation has independent boundary conditions and is disconnected from the global version. That is, solving the pressure equation will not guarantee that the new internal fields are going to be related to the global simulation.

Also, the procedure has to be repeated for each time step, during unsteady simulations. There is no other way to ensure conservative fluxes, because slight deviations from the conservation laws are introduced during each time iteration. An approach like that requires that the simulation is stopped and restarted during each time step, with the present implementation. Also, it is important to remember that solving the pressure equation again also enforces boundedness among the internal fields.

It may also be more practical, for the remote extraction of space-time windows inside HPC facilities, to customise a probing class and dump the necessary information at runtime, in parallel.

Time interpolation has been shown to be plausible, given that certain constraints are satisfied. During an unsteady simulation, the space-time window boundary patches constantly flip their behaviour between inlet and outlet. Because of that, for time interpolation to be possible, the phenomenon from any patch behaving as an inlet for the window, at any moment within the space-time interval, must be either constant or linear. If this condition holds true during the space-time window interval, the approximation errors introduced by the interpolation of the boundary conditions are negligible. If the constraint is not satisfied, time interpolation can not be performed, and all the intermediary time steps must be processed for space-time window information.

The problem of identifying a natural phenomenon which can verify the time interpolation constraint, is a matter of physics, and is still subject to debate. For now, time interpolation has only been proved in a specially crafted test case, with a modified mesh to support the limitations, even though it produces unrealistic phenomena. A finer mesh captures turbulent behaviour at the inlets of the space-time window, making time interpolation unsuccessful. Nevertheless, if it can be applied in practical circumstances, time interpolation amplifies all the benefits that are brought by the new, space-time window reconstruction concept.

The highest level of flow complexity is reached during the ERCOFTAC benchmark. Direct comparisons between the global simulation and the space-time window fields is by far not the best way of validating the reconstruction, but even so it proved that the two simulations are in solid agreement. A quantitative study revealed that the average difference varies from nearby 0 to lower than 0.5%. Also, the total number of cell centres with a difference of at least 5% is less than 0.32% out of the 10^6 points.

These findings support the conclusion that space-time window reconstruction based on submodelling is a functional solution, with promising results, and that it can be further investigated and refined together with experts from hydraulic engineering (mainly CFD), and applied mathematics.

Whenever flexibility can be traded for perfectly accurate field reconstruction, the author believes that the solution based on interprocessor traffic is more appropriate. This method is demonstrated in the next chapter.

9

Space-time window reconstruction in a 3D unsteady complex flow based on interprocessor traffic

This method is designed to reproduce the very same floating point numbers inside the space-time window, using a data storage with intercepted communications, which are recorded during the global simulation run. The idea is engineered as a refinement of all the previous test cases, and is intended to validate the space-time window reconstruction concept as a practical and robust way to deal with large numerical simulation data. The implementation is a particular case in the concept, where all the interpolation errors are removed, in order to obtain the very same fields, with the very same floating point numbers. The price paid for robust accuracy is given by the fact that the new, reconstructed simulation, although independent from the global version in the High Performance Computing facility, does not provide any more flexibility than common submeshing techniques. However, this version is ready to be applied on real-world, industrial simulations, with considerable alleviations to the data deluge problem. The chapter is organised in a demonstrative part, followed by the concluding remarks, as follows.

9.1 Proof of concept

The ERCOFTAC square cylinder benchmark is used, with a mesh of 402144 cells. The case is well known in literature, and is described in Lyn and Rodi [120]. The overall bounding box is given by (-0.4 -0.28 0) (0.96 0.28 0.392), in meters.

After 6 seconds of simulation, the mesh is refined in the region of (0.22 0.06 0.03) (0.29 0.13 0.35). The fields from the originally unrefined simulation are mapped, using linear interpolation, on the refined case.

The mesh is then split in half, with one subdomain described by the bounding box of (0.208 0.0525 0.01) (0.302 0.15 0.382), and one containing the remaining cells of the analysis domain. The first subdomain, belonging to processor PE0, has 671510 cells and includes the refinement region. The submesh of processor PE1 is kept coarse, with 399064 cells, and contains the rest of the global analysis domain. For now, this is regular mesh refinement, with domain decomposition.

In order to intercept the communication between the parallel processing elements, the link channels between the processors are patched with recorders. In this case, these are the processor patches.

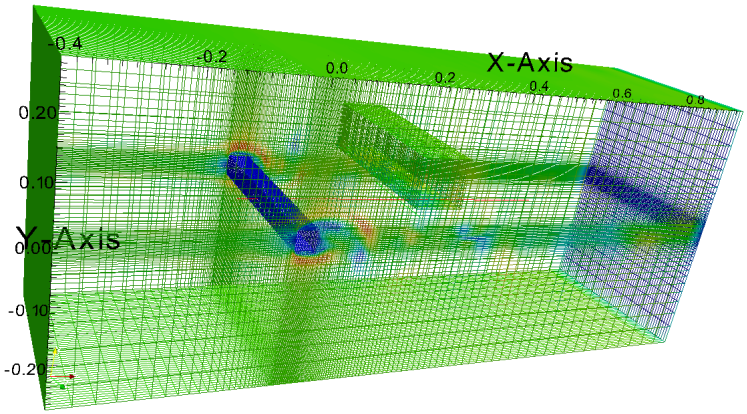


Figure 9.1: The mesh for processor PE1

Fig.9.1 shows the mesh of processor PE1 with an internal window-like subdomain missing, since it is allocated to PE0.

In order to limit the number of communication patches between PE0 and PE1, the mesh of processor PE0 is only refined at kernel level, leaving the peel untouched with 1762 communicating faces, and with direct mapping between the crusting cells of PE0 and those of PE1. Each processor face of PE0 is simply shared with PE1.

Fig.9.2 shows the velocity contours for PE1, and Fig.9.3 shows the corresponding velocity vectors. The flow forms a Karman vortex street behind the square cylinder obstacle. These kind of phenomena have been described in the previous chapter, and are frequent in nature. Because of that, they are also important in the engineering fields, and have always been used for benchmarking numerical analysis software.

The region for processor PE0 can not be seen, being completely internal. It is hidden by the visualisation software.

The original, unrefined simulation covers 10 seconds of analysis. The

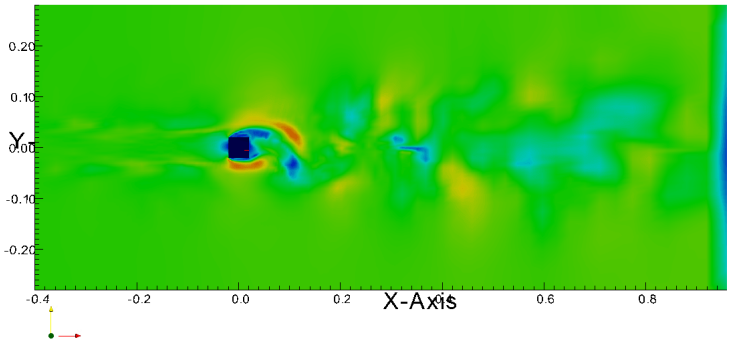


Figure 9.2: Velocity contours for PE1 at $t_1 = 6$ s

space-time window is designed to reconstruct the exact internal fields for one second, from $t_1 = 6$ s to $t_{1000} = 7$ s, inside the subdomain of PE0. In order to capture the space-time window, the refined simulation is run from t_1 to t_{1000} and the complete set of scalars, vectors and tensors, which are communicated during the parallel execution between the two processing elements, are stored in a dedicated archive.

Fig.9.4 displays the evolution of the PE0 data between t_1 and t_{1000}, outlining the global context for better visualisation. The reconstructed data is shown in Fig.9.5. Figs.9.4(a), 9.4(b) and 9.4(c) have a direct correspondence with Figs.9.5(a), 9.5(b) and 9.5(c).

The reconstructed data in PE0, which is based on the interception of

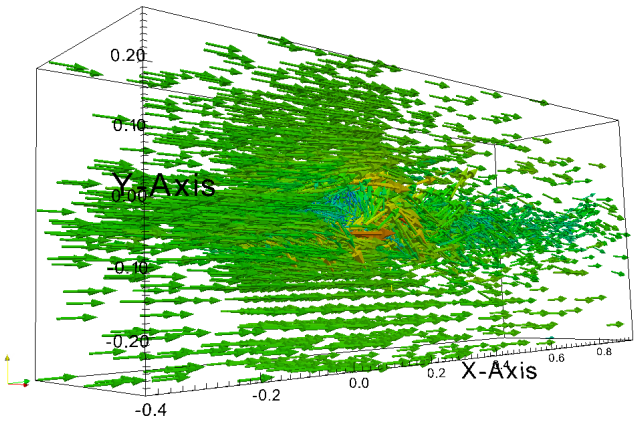


Figure 9.3: Velocity vectors for PE1 at $t_1 = 6$ s

interprocessor traffic, is a bit-by-bit mapping with the original internal fields from PE0 (after the refinement). The solver and the simulation parameters have to remain unchanged, otherwise the reconstruction can not be performed.

A constant time step of $\Delta t = 10^{-3}$ seconds has been used, to speed things up on limited hardware resources. It is a little bit of a border value, with CFL reaching the maximum limit.

The intercepted interprocessor traffic registered 395154 floating point numbers travelling into each of the 1762 processor patches of PE0. On the other hand, a single time step in the refined PE0 region requires about 10.8×10^6 floating point numbers.

Accidental experiences with floating point truncation have been performed. Even the slightest modifications of the interprocessor data can render the process not functional, resulting in divergent solutions or a different number of convergence iterations, which can not be extrapolated using the intercepted traffic.

The problem with this procedure is that it does not provide any more freedom to the user than standard submeshing approaches. However, it can provide substantial reductions of the bottleneck problems.

Fig.9.6 shows the amount of floating point numbers that have been transferred during the simulation of the space-time window period. For the complete second, the average number of 64-bit values is about 700000 per time step.

During the first three time iterations, there is a burst of floating point numbers, which gradually decreases to the normal regime. This is contributed by the initial start-up of the parallel numerical implementation.

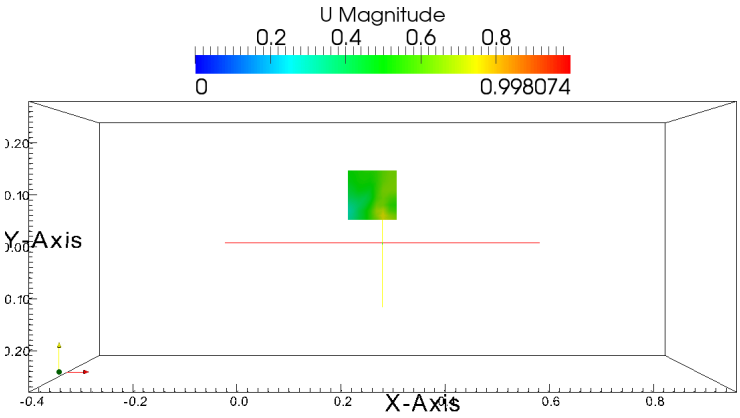
The pattern of variations is pretty much regular, and it is inflicted by the convergence rates, from within each of the individual time steps.

Fig.9.7 shows the relationship between the compression ratio and the number of time steps that get stored on the disk, from the global simulation.

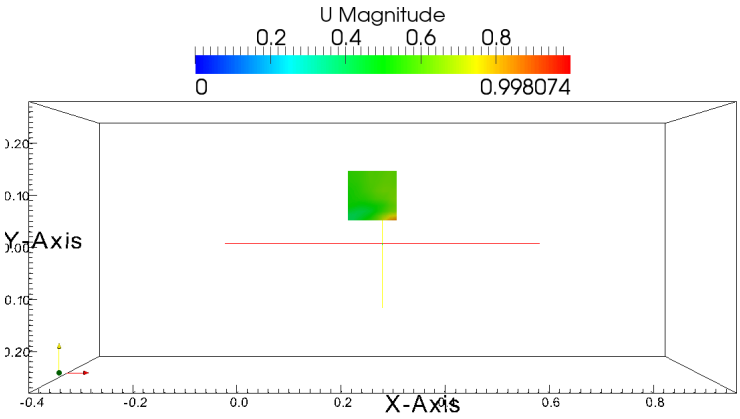
The compression ratio CR is defined by (9.1), where $F_{traffic}$ is the number of floating points counted in the traffic, and $F_{subdomain}$ is the number of floating points that are necessary for storing the fields in the refined PE0 subdomain.

$$CR = \frac{F_{traffic}}{F_{subdomain}} \times 100 \tag{9.1}$$

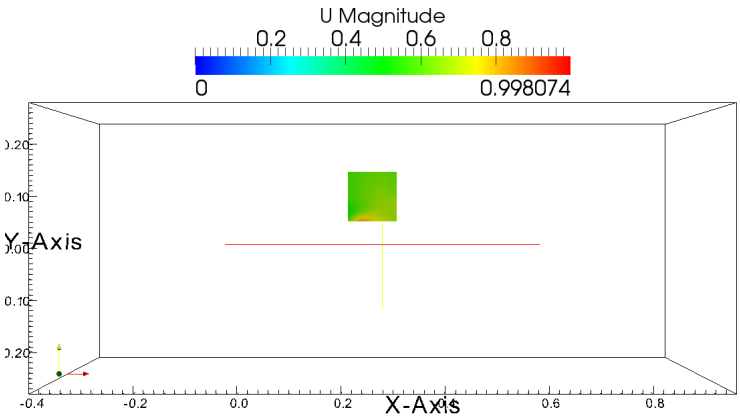
If less than 6.5% of the time steps need to be stored to disk, from the total of 1000 in the space-time window, then the compression ratio is higher than 100%. This literally means that the intercepted traffic overwhelms the number of floating point entries which are required to store the fields in the PE0 subdomain in the first place. But when the number of stored time steps



(a) $t_1 = 6$ s

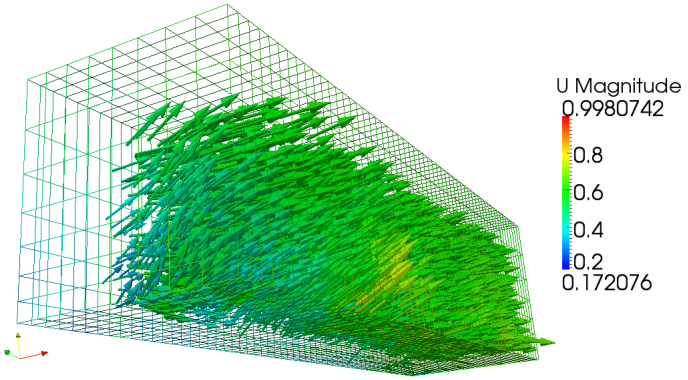


(b) $t_{500} = 6.5$ s

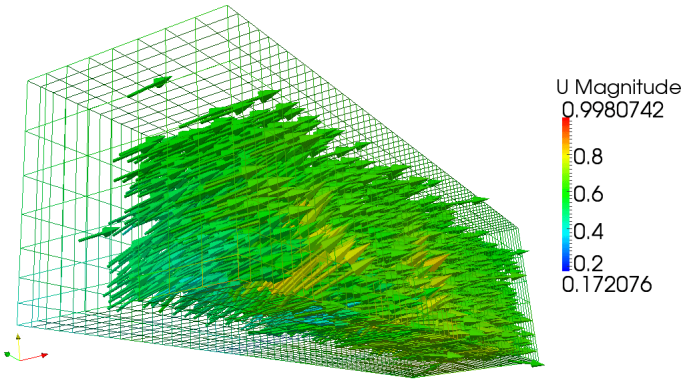


(c) $t_{1000} = 7$ s

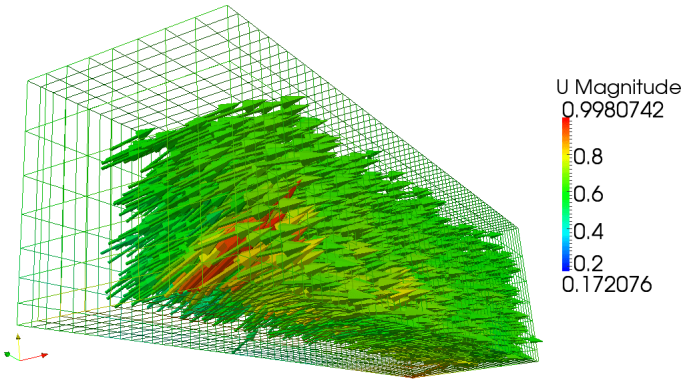
Figure 9.4: Velocity vectors for processor PE0



(a) $t_1 = 6$ s



(b) $t_{500} = 6.5$ s



(c) $t_{1000} = 7$ s

Figure 9.5: Velocity vectors of the reconstructed fields for processor PE0

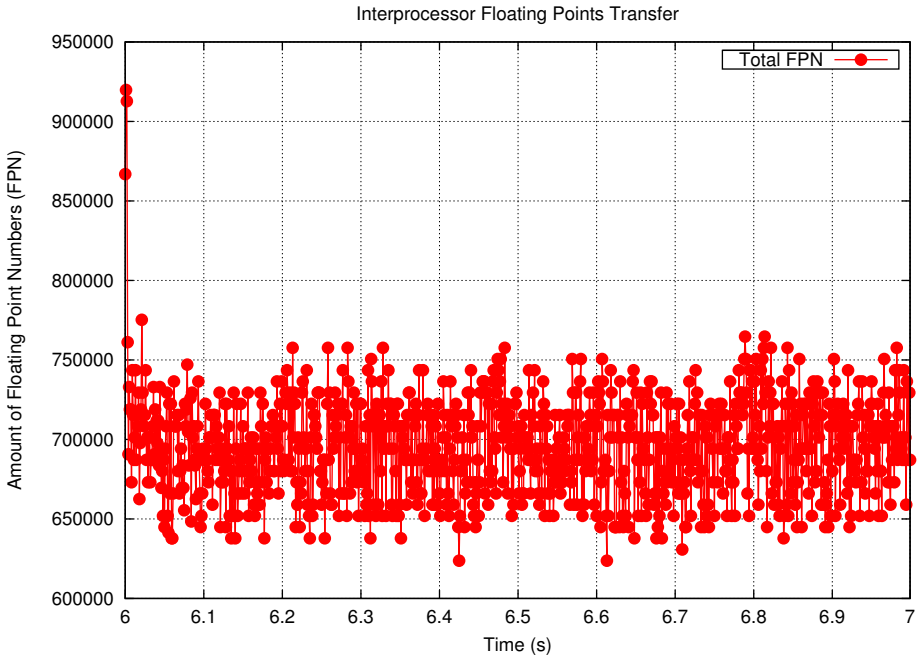


Figure 9.6: 64-bit Floating Point Numbers during the interprocessor transfer

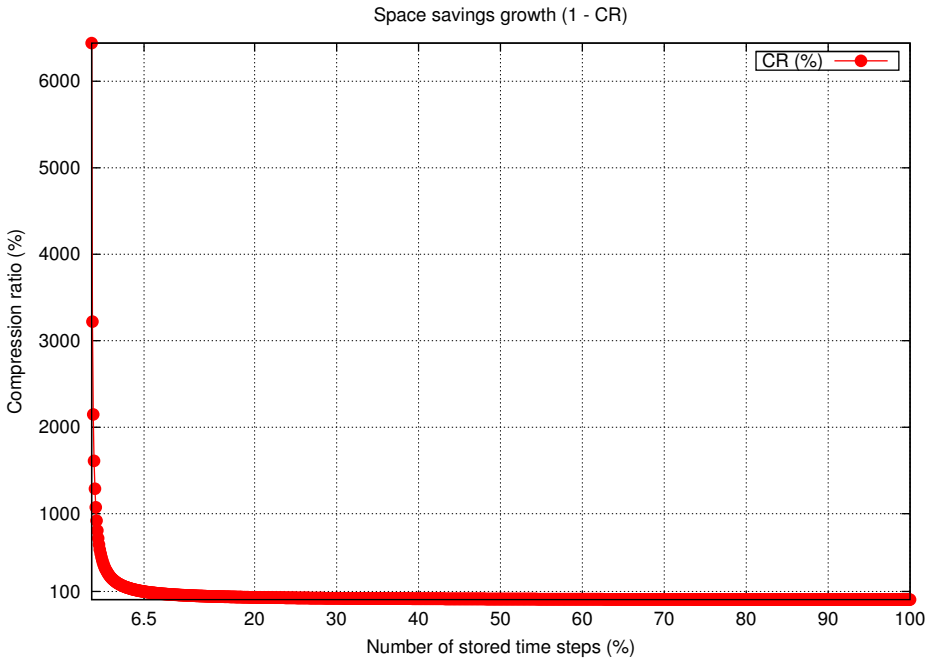


Figure 9.7: Compression Ratio vs Time Step Storage

exceeds 6.5%, the compression ratio becomes practical, being smaller than 100%. The plot reminds of an exponential function with a negative exponent.

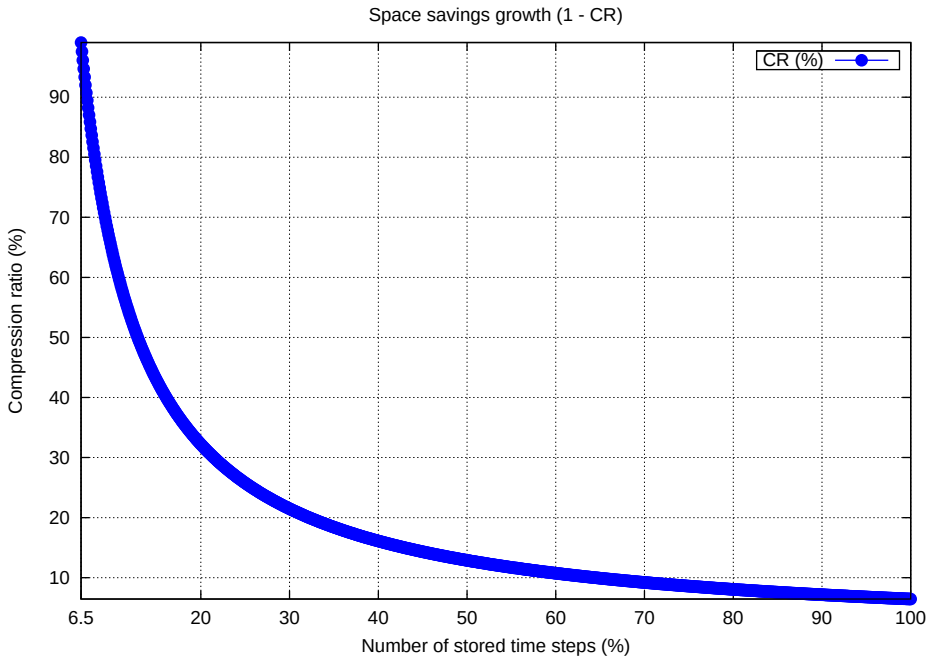


Figure 9.8: Practical Compression Ratio

Fig.9.8 shows the practical region of the compression ratio, on a more appropriate scale. It can be seen that when more than 12.8% of the time steps are required on disk, the compression ratio drops below 50%, increasing the space savings (1 - CR).

Therefore, the space-time window reconstruction based on interprocessor traffic, pays off to be a solid solution to the data deluge problem, with guaranteed, controlled reconstruction of the internal fields, and which can be directly applied in the industry.

Obviously, the entire concept can not be applied unless the user is capable of defining regions of interest, in both time and space, inside the global simulation. Also, for this approach to work, it is mandatory that the regions of interest require local mesh refinement. If these two criteria are met, then the space-time window reconstruction based on interprocessor traffic can be successfully applied.

In order to restart the simulation, one needs to begin with known initial fields, either from a previous time step containing the necessary field values inside the space-time window subdomain, either from global initial conditions.

There are two key factors that control the performance of the solution, in

terms of compression ratio:

- the compression ratio increases with the number of stored original time steps inside the subdomain
- the compression ratio decreases with Δt (smaller time steps produce more interprocessor traffic)

The interprocessor traffic is also affected by the numerical algorithm, and by the number of processor patches that are used for communication. The fewer the communication channels, the better the compression ratio.

It is important that the variations inflicted by the individual convergence rates, during each time step, remain relatively constant and predictable through the space-time interval. Otherwise, an uncertainty factor is introduced, and the amount of interprocessor traffic can be harder to predict, when deciding to apply this space-time window reconstruction solution 'in the wild'.

The concluding remarks discuss the benefits and trade-offs that need to be considered before using this implementation.

9.2 Concluding remarks

The previous chapters were searching for a method to free the user from the supercomputer facilities and from the data bottlenecks, which limit both the way and the place where scientists can make use of large scale numerical data.

This chapter demonstrates a solution, proven reliable, which has a strict focus on lossless data reduction. The very same floating point bits are obtained, but based on less information. The solution uses intercepted interprocessor communications to reconstruct a space-time window, given that such a region can be identified in the global simulation, and that it requires local mesh refinement.

The method can be used as a remote refinement technique which is applied on a subdomain in both time and space. If used properly, according to the blueprints demonstrated in this chapter, the method drastically reduces the data bottlenecks, and practically removes the data deluge problem.

However, it is important to notice that certain criteria have to be met. The compression ratio highly depends on the size of the Δt time step and the number and size of the time steps in the extracted subdomain, that require disk storage. If used improperly, the interprocessor traffic may well overwhelm the size of the original data, making the procedure impractical; large eddy

simulation (LES) usually require smaller time steps, around 10^{-8}–10^{-6} or worse. The smaller the time step, the more traffic that needs to be generated between processing elements.

Another important factor is the numerical algorithm in place, used by the solver. To be more specific, the convergence rate of the algorithm is very important. The sooner it reaches an acceptable solution, the better for the traffic size. Each additional iteration inside the individual time steps creates new floating point exchanges.

The intercepted traffic data needs to be stored by starting with a time step in the global simulation where the internal fields are known. This can either be from the initial, global conditions, or a certain moment in time when the space-time window region starts being of interest.

The number of communication patches between the space-time window processor and the rest of the computational universe, must be reduced to as few as possible. The convergence of the numerical algorithm is also important, because if the blueprints in this chapter are to be used on industrial applications, the variations in the convergence rates must remain relatively constant, and predictable throughout the interesting time interval. Otherwise, the amount of interprocessor traffic may be harder to predict, and thus making the compression ratio much harder to estimate and control.

There are also cases where the method is not applicable, like when data refinement in the interesting space-time region is unnecessary. Also, it may be that the necessary refinement is not dense enough to allow the method to pay off. The best compression ratio is achieved when the local refinement has to be orders of magnitude higher than the coarseness of the 'external' mesh. This is driven by the physics, and the user must create prospects, according to the given blueprints, before applying the procedure.

The current implementation is based on modified processor patch classes. The floating point numbers are dumped in text format, and they have to be converted for compact 64-bit representation. To avoid truncation errors, the precision of the output format must be set to the maximum.

When the data is read, a different, modified version of the processor patch is used. It is important to observe that even the slightest truncation errors, of 10^{-300}, can ruin the reconstruction. Therefore, if any compression is further applied on the traffic data, it has to be lossless.

The number of processing elements that are used during the manual domain decomposition is not important. The more processors it has to communicate with, the less data the space-time window will exchange to each one of them. However for simplicity reasons, it is recommended that only two subdomains are used during the traffic interception procedure. It may even be very hard to define a space-time window in the global subdomain, using

manual decomposition, among another n processing elements. The weight of the computational workloads should preferably be well balanced (although this is not a requirement).

On the other hand, more complex implementations should also be able to describe a space-time window as an entity consisting of multiple arbitrary subdomains, so that the global simulation run can go unhindered, without changes in the domain decomposition, and the necessary traffic is intercepted automatically. The resulting traffic archive can then be used to convert itself to a simple two-processors scenario, by binding the internal subdomains together.

In spite of a radically different implementation, the interprocessor traffic based solution is just a refined validation of the previous test cases and of the 'space-time window reconstruction' concept. All sources of interpolation inaccuracies from the previous experiments are completely removed here. The nature of the processor patches is not that of real boundary conditions, allowing the internal fields to be reconstructed in full precision. If and how these space-time window boundaries based on the interprocessor traffic archive can be converted to a space-time window for solution A, based on submodelling, with unsteady boundaries, is still a matter of research.

The final chapter summarises the thesis and draws the concluding remarks, outlining the main contributions, and presenting possible future research directions.

‘If we wonder often, the gift
of knowledge will come’.
Arapaho tribe

10 Conclusions

10.1 Concluding remarks

The main concern of this study, is to find a solution for the bottlenecks that are caused by the deluge of data, in the supercomputing community. To be more precise, the research targets the numerical simulation community, and is concentrated around a new, proposed concept, called ‘space-time window reconstruction’. The results are summarised and thoroughly discussed in this chapter, followed by the contributions and the perspectives for the future research directions.

The thesis begins by introducing the main motivation behind the research, which is mainly that today’s supercomputers are orders of magnitude faster than our data handling abilities. High Performance Computing (HPC) is compared with a calash where horses have to travel long distances, but dramatically outrun the storage. The specialised facilities are compared with giant aquaria where one can build the most sophisticated submarine technology, but remains constrained to the possibility of gazing at them through the windows. Therefore, the ‘space-time window reconstruction’ concept is synthesized as an investigation area, and sets forth the research objectives for the PhD program.

In Chapter 2, a new perspective for understanding computational science and engineering is presented. The idea is founded on original and unconventional approaches to the literature, including the data mining of ~6000 scientific articles with clustering techniques. The results have been published in Anton [32] and Anton and Creţu [28]. The most important professional societies and the hallmark publications which stand for annual reviews and surveys in computer science, are also remarked. They consist of powerful driving forces and watcher parties, capable of developing and comprehending the vast field of computer science, and leading the computing curricula around the world. The selected publications are useful for identifying the challenges, the margins, and the evolving trends in the branches of computer science.

The chapter is concluded by the proposal of a new perspective on High Performance Computing and the computational sciences[32], with an emphasis on the links that bring the areas of interest from computer science at peering with the necessary fields of science, in order to forge the computing in science and engineering (CiSE) community.

The next one is a thorough and comprehensive synthesis of the state of the art methodologies, covering the solutions for dealing with large numerical simulation data. Different schools of thought are analysed and drilled down to their historical roots, and the most important techniques, from computer science and applied mathematics, are identified. The computational fluid dynamics community is also mentioned, with its 'hands-on' approaches for getting things done, within the limitations of the current framework.

From computer science, specialised peer-2-peer storage systems, and scientific floating point compression are the methods selected as more appropriate for the job. The Freeloader Project developed by Vazhkudai et al. [6] is a hallmark for distributed peer-2-peer systems designed for large simulation data. However, its optimal use is limited to local area networks, and it best fits as a poor man's storage system with institute-level implementations. Scientific floating point compression, on the other hand, can be also used as an offline solution, like it is in the implementation of Burtscher and Ratanaworabhan [7]. However there is a compression ratio wall that limits the effectiveness of the method, and it is partly justified, because the algorithm is generic, rather than optimised for numerical data.

The mathematicians have come up with methods like wavelet compression, submodelling and partial matrix inversion. Wavelets are very good at extracting the most relevant bits of information from a given signal, but they have very poor performance on computational fluid dynamics data. Therefore, they make more sense to be applied when the unimportant part of the signal can be discarded.

The separation of a signal into important and unimportant bits is done by the wavelet, not by the user. Such methods are useful for remote visualisation, as a form of smarter multi-resolution techniques, but not for intelligent CFD post-processing ant bit-level data reconstructions.

Submodelling is a technique one half of a century old, developed in the '60s by structural engineers, working in the aerospace industry. Submodelling, generally speaking, has only been applied on linear phenomena, or at least on linear cut-boundary regions. Most of the times the implementations are limited to steady-state simulations. An essential critique against classic submodelling, is that it has only been applied, where the physics of the studied phenomenon recommends the fact that the local behaviour can be naturally disconnected from the global evolution. Such a thing does not exist

in the middle of a turbulent, unsteady flow, frequently happening in fluid dynamics. This is called the St. Venant's principle and it is well explained in the chapter.

Partial matrix inversion covers a broader spectrum of methods, one of which is substructuring. With partial matrix inversion only a half of the original system of equations needs to be calculated, the other part being related through a direct pre-computed formula. These methods, no matter how optimised, can not be applied on large systems of equations, such as those originating in fine resolution LES meshes.

None of the state of the art approaches seem to take into account that numerical simulation data, no matter how large, is always being produced by a known formula, within a known context. Generally speaking, all these approaches lack the interdisciplinarity that is required to dive deep into the reality of the data deluge problem, and so they only manage to scratch the surface of the issue without any decisive effect.

The 'space-time window reconstruction' concept in Chapter 4 cuts the Gordian knot by drilling down to the root of the problem. The chapter proposes two different solutions for implementing the new concept. The first one is based on submodelling, and concentrates on the flexibility of data post-processing and re-processing. The second one concentrates on data reduction and provides lossless reconstruction of the internal fields, with bit-level accuracy. Basically, Solution B is a particular case for Solution A, but with interpolation inaccuracies removed.

The first solution has been disseminated with many occasions, like in Anton [122], Anton and Creţu [123], Anton [130], Anton and Creţu [131]. The whole idea is based on the observation that most of the times numerical simulations, including the computational fluid dynamics ones, require a large domain of analysis to be calculated, in both time and space, when only a part of it provides the information that is being sought for. For instance, the vortex shedding that is happening behind the cylinder obstacle in the ERCOFTAC benchmark (Chapter 8), is symmetric against the X-axis. This makes it unnecessary to retain or transport the mirror data, since it can be obtained based on the other half of the simulation domain. Also, the numerics require from 6 to 10 start-up seconds before beginning to produce reliable data which can be contrasted with laser made measurements. This suggests that an entire period in simulation time can be dumped, and that it should not be transported or stored. Such examples bring up the idea that it is more practical to select one or more user-defined space-time regions in the global simulation domain. These are called 'space-time windows'. But there is a high probability that the mesh in the user-defined region has been customised and refined, and with large eddy simulation data the information in space

and time that needs to be extracted is still too big.

Therefore, the submodelling based solution tries to extract the minimal information in that region and use the computational power available on end-user machines to recalculate the internal fields inside the space-time window. This means that besides the initial conditions, the space-time window must retain the necessary boundary information for each of the reconstructed time steps. All this data is stored in a space-time 'capsule' format, and later serves as the basis for configuring the reconstruction case. For now this is done by using the same solver, but different solvers can already be tested.

The problem with this solution is that the reconstructed flow can be slightly shifted in both time and space. The best way to control it is to use reference data, preferably experimental, for calibration. Such data can be obtained with laser equipment and pressure sensors. To best emphasise the user interest in the space-time region, consider that traditionally, in most real-world applications with turbomachinery, controllable data has only been known at the inlet (velocity, mostly constant) and at the outlet (pressure). What happens inside the 'black-box' geometry can only be 'seen' with the computer, through numerical analysis and visualisation, and nowadays partly with modern equipment. Even when laboratory installations permit it, the costs and the effort are prohibitive. The submodelling based solution brings unprecedented flexibility to the CFD user.

The second solution is using interprocessor communications to reconstruct the space-time window. When the numerical analysis reaches the point in time where the space-time window begins, the domain is decomposed so that one of the subdomains contains the region. For the method to be efficient, the data inside the kernel of the space-time window has to be refined (if it is not already fine enough), so that the crust of the subdomain, consisting of processor boundary surfaces, is much coarser. The decomposed simulation is continued with parallel execution, and the traffic between the processor holding the space-time window and the rest of the computational world (usually another processing element for the remaining of the mesh) is intercepted and stored. The internal fields inside the space-time window are then reconstructed using the traffic archive and the same solver. This idea is in fact a particular version of the first, but one that can be implemented by removing all the interpolation inaccuracies from the space-time data.

When dealing with petascale or larger amounts of numerical simulation data, the simple extraction of the fields in a space-time region can still mean too much information to handle with present-day communications and storage technologies. Also, considering that the space-time region is of high user interest, with large scale simulations it is highly probable that the mesh inside it is going to be extremely fine. In this scenario, the 'space-time window

reconstruction' concept is much recommended. If the very same fields need to be obtained without any post-processing, the second solution can be used for optimal compression ratios. When the data has been reconstructed on the user-side, the first solution can still be used for flexibility (for instance, for changing the solver). Therefore, the concept proposed in this chapter is a compact, single and robust stand-alone solution to the numerical data deluge problem.

As a standard procedure, the author recommends that the user applies the 'space-time window reconstruction concept' in two parts, as follows: apply method B to bring the data on personal hardware, and method A for secondary post-processing of the data, after it has been reconstructed with method B. This two-part procedure ensures that no matter what happens with the information inside the data centre or within the supercomputer, the important part of the simulation is always backed up and available with bit-level accuracy.

Since the new concept is a bleeding edge idea in CFD, the research requires a well-defined strategy, with many levels of chariness. The research strategy is condensed in Chapter 5.

The whole idea is split-up into problems of incremental difficulty. Like any other engineering field, computational fluid dynamics is an art of approximation. According to Muntean [119], there are three major types of approximation in CFD:

1. Temporal approximation, where the ΔT time step range is selected, and time-dependant variables and coefficients are identified;

2. Spatial approximation, which defines the number of spatial variables used in the model;

3. Dynamic approximation, where the equation terms which have a negligible impact are removed, so that the computational resources are exploited to the maximum;

At the first level of complexity, a laminar, non-dimensional problem is used, for a 'proof of concept'. The non-dimensionality is given by the stream function, which can be used to derive the velocity fields in 2D. When a new road is opened and there is no previous work on the subject, there always has to be 'a first'. Regardless of how trivial it looks, 'the first' always rises serious and unsuspected challenges and may be harder to put down, therefore it is better to keep it as simple as possible and verify if the idea can be done. This is the sole purpose of Chapter 6 where the 'proof of concept' is explained.

The next level basically moves the problem from the finite element method (FEM) to the finite volume method (FVM), and introduces more complex, rotationary flows. The FVM is more recent and frequently used in the CFD community; but the FVM with cell-centred values brings up a serious problem: one can not define a cut-boundary extraction region, in such a way as to remove interpolation completely. This, on the other hand, is possible with the FEM, because the calculated values are stored in the nodes of the mesh, not in the centre of the elemental volumes. There is no workaround for this issue, but this is where the second solution serves as a refined validation of the concept (Chapter 9).

Due to the cell-centred FVM problem, from now on, throughout Chapters 7 and 8, a level of uncertainty is always present, and the physical laws, with the conservation of mass, are used to monitor the quality of the interpolation. The problem at this level is bidimensional, but the OpenFOAM implementation claims for cvasi-3D.

The ultimate level of complexity is reached in Chapter 8, where the CFD approximation is limited to the dynamics of the system; and even in this case, a large eddy simulation (LES) model is used, which is complex and computationally intensive.

The research strategy also includes a backup plan. At any point in the roadmap, a failure to obtain the 'space-time window reconstruction' would mean that the submodelling based solution is not suitable for the goals of the investigation, and that a new idea is necessary. If that was to happen, plan B would have meant to limit the investigations to solution B, which is 'safer'. Fortunately this was not necessary, and the combined solutions of A and B provide a very solid remedy to the numerical data deluge problem and a compact, powerful two-part procedure for the HPC community. The solution in plan B is independently studied in Chapter 9, as a refined validation of the previous work.

Chapter 6 serves as the 'proof of concept' case, where the idea is demonstrated for the first time, using a trivial computational fluid dynamics (CFD) problem. The implementation is completely based on the finite element method and the PETSc Toolkit developed by Balay et al. [26]. The test-case is from Resiga et al. [109]. The first and most important thing that is learned during this stage, is that the 'space-time window reconstruction' concept, although merely only a spatial reconstruction at this time, is possible, and that it can be applied to CFD. The author discovers the classic submodelling method from structural engineering much later, in Chapter 8, after all the complexities have been tackled with.

Another important step is the study of how interpolation algorithms can affect the accuracy of the reconstruction. The tests are performed with the

help of the GSL library[126]. The conclusion is that linear interpolation is good enough, and based on that, linear interpolation is used throughout the remaining of the thesis, and in all the experimental scenarios. Last but not least, without prior knowledge of the classic submodelling method from structural analysis, the author experiments with different mesh resolutions, understanding that the problem of 'space-time window reconstruction' is more than just a problem of data extraction, data reduction and reconstruction.

Chapter 7 bears the fruits of joining an open source CFD community, and is mainly focused on the implementation issues that have to be resolved before teasing the tough problems in OpenFOAM. The test-case also proves that the boundary extraction method can handle rotationary phenomena, where vortices are forcing the flow to get back into the reconstructed area.

Chapter 8 reaches the highest level of complexity, in terms of the modelled phenomenon. Most importantly, this is where turbulence and physical time are introduced. The chapter is organised in three sections and two major test-cases. The first section deals with the implementation of the submodelling based solution, which is now solely based on the OpenFOAM toolkit and original software.

The first test-case is specially crafted for time interpolation. The importance of time interpolation for the 'space-time window reconstruction' concept is major. Any time step that can be interpolated does not require storage, and therefore can be removed from the space-time archive. However, the mesh of this specially crafted test case has been altered such that it does not capture real, natural phenomena. The diesel-injection cavity problem is used to see if time interpolation is possible at all, and if successful, to understand the constraints that need to be taken into account.

The first observation is that in an unsteady simulation of turbulent phenomena, the boundary patches around the space-time window exhibit inlet-outlet flipping behaviour, during each time step. This can mean tens of thousands of times per second, or much more. The problem with turbulence is that whenever nonlinear phenomena happens around the cut-boundary region of the space-time window, any single inlet can prevent time interpolation from being possible. Time interpolation is shown to be possible as long as the nonlinear phenomena are constrained, during each time step, at the boundary patches which manifest outlet behaviour.

Finding a natural phenomenon which is suitable for 'space-time window reconstruction' with time interpolation, is a problem of physics, and is still a matter of debate. It may very hard to find a natural scenario for this one. On the other hand, time interpolation amplifies all of the benefits that occur when applying the 'space-time window reconstruction' concept which is based on submodelling – in this case, 99% of the time steps do not require storage in

the space-time window. The time interpolation can be performed, as long as the phenomena at the cut-boundary region of the space-time window displays constant or linear evolution through time for each of the boundary patches which behave as inlets. Such a case is demonstrated with a specially crafted mesh. However, a mesh which captures the natural phenomenon correctly produces turbulences at the main inlet zone in the cut-boundary region of the space-time window, and this makes the time interpolation attempts unsuccessful.

The ERCOFTAC square cylinder benchmark is well known in literature – see Lyn and Rodi [120] – and it is widely used for the validation of numerical software. The turbulent flow over a cylinder produces a Karman vortex street, with vortex shedding behind the cylindrical obstacle.

In nature, Karman vortex streets are frequent. Even if invisible to the human eye, they can form when the wind crosses over the electricity lines, producing vibrations and sometimes a whistling sound. They also form in the sky around high-altitude mountain peaks or in the ocean, around the islands that are blocking the flow of the ocean streams. A frequent encounter is around industrial chimneys, where they induce structural stress and can provoke disasters. In order to attenuate the structural stress, spiral ridges are attached around the chimneys. Square cylinders are building blocks for many engineering applications, under water, in the sky or in other fluids, therefore the studied phenomenon is of real practical interest.

The 'space-time window reconstruction concept' is used to reconstruct a space-time window containing the flow right behind the square cylinder. The vortex formations are emphasised with the help of pressure isosurfaces. The same flow features are reconstructed successfully. Global and reconstructed streamlines are also compared from different angles, to prove that the same flow features are preserved. However, for a more convincing comparison, the velocity fields are compared head-to-head.

A direct comparison of the field values between two independent simulations is not the best choice, because the fields can be shifted in both time and space, leading to erroneous conclusions. However, in this case, the comparison brought up solid proofs that the very same data is obtained, in spite of the different mesh resolutions and the possible zoom-in effects. The quantitative study revealed that the average difference varies between nearby 0 to lower than 0.5%. Also, the total number of cell centres with a difference of minimum 5% is less than 0.32% out of the 10^6 points. The results are well explained in Chapter 8.

The final test-case, in Chapter 9, is an attempt to reconstruct the very same floating point numbers, with the very same bits, by using manual domain decomposition to describe the space-time window, and by intercepting the

interprocessor traffic, for storage. The fields are then correctly reconstructed using the traffic archive, which, when used properly, can reduce the data throughput to very low levels. For instance, if at least 12.8% of the time steps within the space-time window region are normally required to be stored on disk, the achieved compression ratio starts dropping below 50%, which allows for impressive space savings. The key point is to refine the kernel of the space-time window as much as possible, while maintaining the communication peel untouched, and as coarse as it can be kept.

The 'space-time window reconstruction' concept is based on the fact that, in regular numerical simulations, and especially in computational fluid dynamics, only a subpart of the global analysis domain, in both time and space, contains useful and targeted information. The author demonstrates the concept in test-cases with gradual complexity levels, starting from ground zero, and finally ending up by laying down the foundations for a new style of producing numerical simulations – with immediate applications in CFD.

Fig.10.1 shows the evolution for the individual processing power of the processors used in the Top500[12]. It is based on unofficially collected system data. As shown in Chapter 2, the dominant architecture in the Top500 is the cluster, covering more than 80% of the systems. Considering that most of the clusters are made up of computer systems based on regular microprocessors, this implies that similar, or the very same models of microprocessors are available for the end-user hardware. Therefore, the computational power of the hardware which is available to the end-user has been increasing at an exponential rate.

Nowadays, this computational power can also be accelerated with affordable GPGPU and FPGA hardware. With such an increase in the end-user's ability to perform numerical computations on personal hardware, the rationale is to enable the 'space-time window' concept to exploit it, and trade-off personal computational resources for much better space savings and flexibility. In the present research, in the most complex test cases of Chapter 8 and 9 (mainly the ERCOFTAC square cylinder simulation), the reconstruction of the space-time window requires 40-60 seconds of computation per time step, depending on the mesh resolution, which varies from very coarse to very fine meshes with millions of cells. The commodity machine that was used is a Q6600 quad-core running at 2.5 Ghz, and capable of 24 GFLOPS according to the Top500 benchmark, developed by Dongarra et al. [50].

The results in data reduction are remarkable, and surpass the existing state of the art techniques by having smaller data sets. However, the proposed methods are not so generic and can only be useful when certain conditions are met, while others, like floating point compression, can be applied regardless of these limitations.

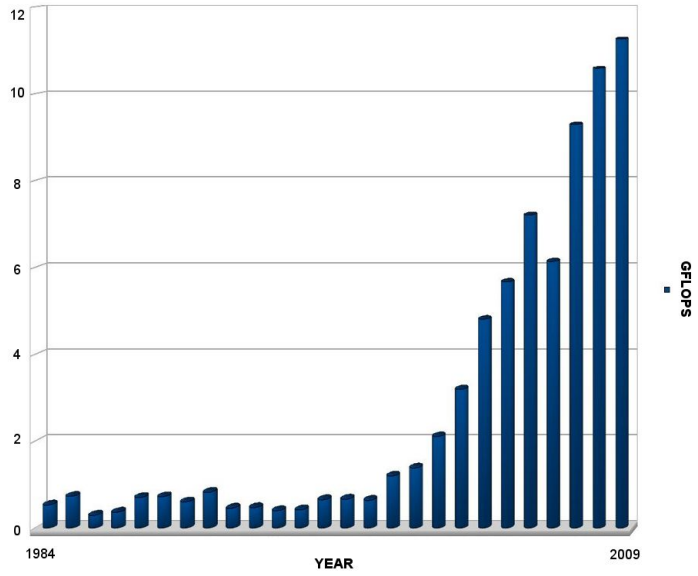


Figure 10.1: Performance of Individual Processors in the Top500

One of the main difficulties is to have a well-posed mathematical problem. For instance, during the reconstruction process, after each new time step is enforced with boundary conditions, the velocity fields need to be normalised against the interpolation error, such that mass conservation is imposed by artificial, iterative adjustments.

After the fields are normalised, the fluxes need to be recalculated using the pressure equation, as described in Jasak [78]. The author has developed this procedure, but it is not enough. It has to be applied during each new time step, before the actual calculations begin. This makes the process tedious and slower, and future investigations need to be performed. The tests show that even if the fluxes are conservative and recalculated during each time iteration, the process modifies the internal fields, most notably the pressure. So even if one gets perfectly posed problems, it is still a matter of research to see how the side effects are going to influence the reconstructed flow features, which are basically made out of a new simulation. For now, the reconstruction is performed without these options.

The procedure of [78] is also good for rebinding the equations together. The impossibility of flawless interpolation forces the tiny changes that are monitored through the conservation of mass criterion, to break the boundedness of the system. If not fixed, this can hang or crash the implementation in the future, for a large number of reconstructed time steps, or produce divergent solutions.

The next thing to take into account, is that based on the current implementation, the simulation of the reconstruction would need to be started, stopped, corrected and restarted during each time iteration. This is especially difficult while the Δt time step can be very, very small (10^{-8}).

A small Δt means that unless time interpolation can be used, more data needs to be sampled into the space-time window. For solution B, smaller time steps mean more traffic and larger traffic archives. The large eddy simulation (LES) solver that is used normally requires 10^{-8}–10^{-4} time steps, but smaller values are possible.

Another thing to consider is what to expect from a 'space-time window reconstruction'. If the second method is applied, the very same field values are guaranteed to be obtained every time, as long as the simulation conditions are not changed. This means that the very same mesh, with the same solver and the very same solver parameters need to be applied. That includes the domain decomposition during the interception of space-time window traffic.

For the first method, based on submodelling, the validation process is more tedious. Besides a flow feature validation and quantitative comparisons which can be obtained in computer science, in hydraulic engineering, and more importantly in computational fluid dynamics (CFD), validations are much harder to obtain. For instance, the real time in the simulation is irrelevant, unless it is correlated with a specific signal from the experimental data, like a pressure threshold. In Lyn and Rodi [120] the variation of the pressure field right behind the cylinder is measured with a pressure transducer. The shedding of the streamlines is detected by a spike in the pressure amplitude, which triggers the beginning of phase 1. There is no correlation with physical time, but velocity measurements are averaged according to the phase of the sinusoidal signal, which is formed by the fluctuations of the pressure probe.

The numerics need some 'time' to produce stable results. For the square cylinder test case, 6 seconds of the simulation time are used to stabilise the solver. If one would want to compare the simulation with experimental data, the first 6 seconds can be simply discarded; next, some global hydraulic coefficients need to be verified. The simulated phenomenon can also be shifted in space, which means that flow regions too close to the space-time window boundaries are bogus and should not be used. Again, this explains why CFD users frequently have to analyse much larger analysis domains, with longer time intervals, to finally obtain a valid, smaller region of interest, that can be isolated in time and space. Only after the global coefficients are confirmed, the validation goes on with more advanced measurements, like averaged velocity data sampled on the unit vectors. If one has to fit the fields obtained by numerical simulation with data from experimental measurements, the time used in the numerical simulation needs to be synchronised with the phase

bins, in a similar way the experimental measurements are triggered. Simply doing that for large-scale data, like in LES, can be a nightmare. The more time steps are subsampled, the worse the nightmare gets. Missing time steps in the global simulation have to be recalculated (on a very fine mesh), until one of them can be identified as belonging to the beginning of phase 1. They usually also eat up a lot of storage space.

To conclude the submodelling based 'space-time window reconstruction' is, in fact, a brand new simulation, with new boundary conditions, and the reconstructed phenomenon can be shifted in time and space; so it takes a CFD expert to analyse this problem and find a hydraulic relationship, when possible, between the global field values and the reconstructed version.

Small time steps also mean trouble for the second solution, which is based on the interprocessor traffic. The more time steps that need to be calculated in the global simulation inside the time interval of the space-time window, the more data that needs to be intercepted and archived on the disk.

All things being said, the 'space-time window reconstruction' concept is compatible with the main state of the art approaches, and can be used, at any time, combined with methods like scientific floating point compression[7] and peer-2-peer specialised filesystems[6].

The next section summarises the main contributions of the thesis, taking into account that this is an approach which focuses on computer science, with practical and applied solutions.

10.2 Contributions

According to Allen Newell[134], ACM Turing Award winner, computer science PhD theses must have at least one of the characteristics listed in Table 10.1 – see Kung [10].

This thesis has fulfilled four requirements, as shown in Table 10.1. The most definitive contribution is the opening of a new area. Basically, in this thesis the author lays down the foundations for a paradigm shift in the way large numerical simulations are handled. This is done by introducing the new, 'space-time window reconstruction' concept, which has been demonstrated and validated at gradual levels of complexity.

In order to demonstrate the new concept, the author develops new methodologies and also develops new tools, which have been shown to produce reliable results.

The foundations for the new space-time window reconstruction concept are moulded on an extensive study covering different schools of thought. The author thoroughly explores the area of large numerical simulation data

- Opens up new area
  - Provides unifying framework
  - Resolves long-standing question
- Thoroughly explores an area
  - Contradicts existing knowledge
  - Experimentally validates theory
  - Produces an ambitious system
  - Provides empirical data
  - Derives superior algorithms
- Develops new methodology
- Develops a new tool
  - Produces a negative result

Table 10.1: Characteristics of a PhD Thesis in Computer Science – from Allen Newell, published in Kung [10]

handling, surveying a large spectrum of approaches that have attempted to, or could have been used for, dealing with supercomputing data bottlenecks. The best solutions from three main borderline fields are examined: computer science, applied mathematics and computational fluid dynamics.

The 'space time window reconstruction' concept can be used to alleviate the deluge of numerical information to what is minimal, essential data. The main focus has been concentrated on the data size problem, being mainly a computer science challenge.

The author provides two fully working solutions for implementing the 'space-time window reconstruction' concept, both adjusted to different user needs, and both resolving the bottlenecks that appear during large scale data handling in High Performance Computing.

It has recently become more and more clear that a radical change in the way we manage very large scale numerical simulations is necessary. It has also become clear that the beating of more horse power out of the bleeding edge supercomputers hits the data wall before the start of the race. This thesis is an echo of the problem, digging and opening up very deep mines, at the frontier of multiple scientific fields. The new concept could have been approached by experts from applied mathematics, computational fluid dynamics, or any other computational science and engineering, frontier field.

Researchers from other, related fields of study, can use the 'booth' that has been introduced and validated in this thesis, in order to help shape the new paradigm for the future.

Computers have become the third pillar of scientific progress, along with theoretical investigations and experiments. Therefore, the proposed solutions

are engineering the foundation stone for a new way of doing science.

The main research has been centred around the problem of numerical data deluge, which creates bottlenecks for the supercomputing communities. In short, this research has been engineered from the beginning as an attempt to cut the Gordian knot of the problem. The roadmap is explained in Chapter 5. The great bet in this concentrated effort has been to break the ice and open up new roads, then show that they can lead to a radical paradigm shift, in the way large numerical simulation results are handled, bringing serious benefits and resolutions of the data bottlenecks. This is only possible when sailing in uncharted territories, exploring frontiers that have never been challenged before. Therefore, the results in this thesis are of high impact and great potential for computational science and engineering.

The new concept is called 'space-time window reconstruction', and creates a paradigm shift in numerical simulations. It has never been attempted in computational fluid dynamics before, and involves much more complex phenomena than what classic submodelling, from structural engineering, has been designed for. A summary of the primary contributions introduced through the 'space-time window reconstruction' concept is listed below:

- a standard two-part procedure for removing the deluge of numerical data through the 'space-time window reconstruction' concept:
  - the author recommends that the user applies the 'space-time window reconstruction concept' in two parts, as follows: apply method B to bring the data on personal hardware, and method A for secondary post-processing of the data, after it has been reconstructed with method B;
  - this two-part procedure ensures that no matter what happens with the information inside the data centre or within the supercomputer, the important part of the simulation is always backed up, portable, and available with bit-level accuracy;
- a robust method for reducing storage requirements:
  - the data deluge problem is practically solved, due to the fact that the user can now focus on the essential parts of the simulation – see Anton [130], and Anton and Creţu [131];
  - the data bottlenecks are alleviated at all levels, since only a fraction from the global information is necessary for the reconstruction of the space-time window;
  - less data needs to be transported;

- a real solution for alleviating the data bottlenecks:
  - client level, by reducing the amount of data which has to be transported over the Internet;
  - gateway level, by consuming only a fraction of the available bandwidth which connects it with the data centre;
  - High Performance Computing facility level, by relieving the computing nodes from the internal network of the burdening transfers that normally converge in one exit point, during the offloading procedures;
  - the methods in the state of the art concentrate on peer-to-peer technologies, which do not reduce the data, floating point compression, which is too generic and limited in terms of compression ratio, and data reduction techniques which either do not focus on preserving the useful bits from the simulation, either remove the field information along the way replacing it with prefabricated, post-processed data – see Anton [122], and Anton and Creţu [123];
- implementations which adapt to practical user needs, and solve real problems:
  - solution A empowers the user with freedom and flexibility, such as when the space-time window is used for deeper investigations;
  - solution B can reproduce the internal fields in the space-time window by recalculation, with bit level accuracy;
  - solutions A and B can be combined;
- a method for assuring complete data mobility for ultra-scale numerical simulations:
  - stuffing the essential bits of useful information through the eye of a space-time window, after being remotely extracted from a large-scale simulation, enables the user to store and transport that information with ease;
  - the space-time window data can find room on any commodity technology available to the end-user: microSD cards, USB sticks and gadgets, mobile phones and netbooks;
- a method which frees the user from depending on expensive and/or scarce hardware:

  - the hardware used in High Performance Computing facilities is specialised and expensive, for both the computational purposes and for storage, bounding the user to it whenever large-scale numerical simulations are performed;
  - the space-time window can be reconstructed using affordable acceleration techniques, like General-Purpose computation on the Graphics Processing Unit and field-programmable gate array boards – Anton [130];

- a new method for performing zoom-in and zoom-out effects in the middle of real-world, complex computational fluid dynamics simulations:
  - the method can be used as a virtual periscope which can be harnessed to move around and study the data in the global simulation at different resolution levels, even while being outside of it – Anton [122];

- an implementation which brings flexibility and independence in how the results are post-processed and re-processed:
  - the user can change the dynamics model inside the space-time window if a different one is more appropriate, as long as the necessary fields are available;
  - the mesh inside the space-time window can be refined and modified in any way – Anton and Creţu [123];
  - a different numerical solver can be used, even for the same mathematical model;
  - other space-time windows can be recursively defined inside the first one, if it may reveal useful information about the physics;
  - the space-time window can be used as input or in relation with other numerical simulations, like, for instance, in the case of fluid-structure interactions;

- a solution with a broad spectrum of immediate applications in electrotechnics, physics, magnetohydrodynamics, and virtually any numerical field of science:
  - the space-time window reconstruction concept can be applied in any field of science or engineering which requires numerical analysis, whenever the physics of the simulated phenomenon recommends that a space-time region can be identified in the global simulation;

  - the tools that have been developed can be used to apply the space-time window reconstruction concept in any of the scientific fields that can be approached with the OpenFOAM toolkit[27], either for direct and robust data reduction, either for customising the solution for the peculiarities of the physics;

- a 'software booth' for investigating the new paradigm with experts from other fields, like hydraulic engineering and applied mathematics:

  - the software and methodology that have been developed, just like the 'Egg of Columbus', enable the hard work to be fulfilled (i.e. 'cooked') with interdisciplinary research, that also involves experts from other scientific and engineering domains;

- a method and software tool for submodelling within unsteady turbulent flows:

  - unlike classic submodelling which is regularly used in structural engineering with linear phenomena around the cut-boundary regions, and is seldom used with unsteady processes like material melting, the current method and implementation can be used in the middle of unsteady turbulent and rotationary flows, capturing a volume of fluid in the space-time window – see Anton [130], and Anton and Creţu [131];

- an alternative to compression:

  - the 'space-time window reconstruction' concept can be used as an alternative to compression, with a direct purpose of better compression ratios when the solution based on interprocessor traffic can be used;
  - if obtaining the very same bits for the internal fields is not among the purposes, the method based on submodelling can also be used for compression as demonstrated in Anton [122]

- a method for in-situ processing and local post-processing:

  - the introduced concept is designed to allow for the extraction of the space-time window on a remote basis, and enable the internal fields to be independently recalculated on local, commodity hardware;

- a solution to allow for local changes in the modelled phenomena to be introduced independently:

  - using the implementation based on submodelling, the user can describe moving meshes, mesh deformations, or simply add new elements inside the space-time window, like new or fewer obstacles;

- a solution for fast exchange of simulation results (via Internet, personal storage devices, etc):

  - researchers in scientific and engineering fields can now make use of the space-time window reconstruction concept in order to decouple and exchange numerical information, which could not have been possible before;

- a method for fast and focused post-processing:

  - the tools that are developed allow the user to quickly jump and focus in the interesting space-time regions of the simulation domain, without having the process all the information (remotely or by downloading it) – see Anton and Creţu [123], and Anton [122];
  - the user can now capture and manipulate the essential physical processes, like for instance vortex shedding;

- a method for estimating the finite volume interpolation errors, based on the preservation of conservation laws:

  - for the 'space-time window reconstruction' concept based on submodelling, the accuracy of the interpolation, during each time step, is verified using the physical laws of nature, and mainly through the conservation of mass in the space-time window region – see Anton [130] and Anton and Creţu [131];

Other contributions from this thesis:

- a thorough and profound exploration of the state of the art in dealing with large numerical simulation data:

  - the study covers a very broad spectrum of ideas, diving into independent and different schools of thought, in order to find the best methods for dealing with large numerical simulation data;
  - from the computer science school, the author identifies floating point compression[7] and specialised peer-to-peer distributed systems as outstanding but insufficient solutions[6]

  - from the applied mathematics school, the research covers wavelet compression, partial matrix inversion and classic submodelling techniques, with pros and cons;
  - the computational fluid dynamics school is mined for smart workarounds in data handling, like in-situ post-processing and visualisation;

- an original introduction to 'High Performance Computing' and 'Computational Science and Engineering':
  - the author uses unconventional methods for developing a well formed vision over the evolution of supercomputing, with prospects for the future trends and applications – see Anton and Creţu [28], and Anton [32];
  - using statistical, economical, historical and political data, the thesis presents an in-depth study of supercomputing and the future developments in Europe – details published in Anton [21];
  - the emergent, young field of CiSE is described with an original perspective over the fields of study that are implied – Anton [32];

Many of these results have been published or are being prepared for journal submission. A broader list of publications, covering the aforementioned contributions, is presented in the appendix of the thesis.

The space-time window concept has been engineered for long term development and polishing, as a new style in High Performance Computing. It is also a leap forward for the state of the art methods, and improves the research abilities in hydraulic turbomachinery, and green energy installations. Some of the future research directions are presented in the final section.

10.3 Future research directions

The method has a lot of potential for the future. For instance, feature tracking can be implemented such that the space-time window follows the vortices it reconstructs across the global domain of analysis. The same procedure could be applied for tracking air bubbles during the simulation of laser surgeries.

As a way to improve performance and regulate the reconstruction process, multiple space-time windows can be defined over the same spatial domain, but with adjacent time intervals.

The interpolation used during the extraction of the space-time window can be highly improved, using intelligent sampling positions that follow the information density inside the global data.

In order to normalise the boundary conditions, a self-healing method should be implemented as a stand-alone boundary condition which is capable of auto-adjusting the values during each time iteration.

Real-time extraction mechanisms can be developed based on the probe class, that are capable of dumping out the space-time window directly from a parallel numerical simulation at runtime.

For the extraction process, the interpolation could be improved by replacing the standard cubic grid with a cloud of coordinates which contain cell centres from the global mesh. This can reduce the number of interpolation levels to only one.

In spite of the OpenFOAM [27] implementation and design, it is necessary to study the space-time window reconstruction process when accelerated with GPGPU devices. A key problem to be taken into account regarding this matter, is the lack of error correcting codes (ECCs), in the commodity graphic cards' Random Access Memory (RAM).

As a refinement, the energy losses inside the space-time window could be verified, along with the conservation of mass.

For now Dirichlet boundary conditions have been used. Preliminary tests show that mixed Dirichlet-Newmann conditions can be used when configuring the temporal behaviour of the boundary patches; however, further investigations are necessary, in order to compare them with fully prescribed boundary values. Also, with mixed boundary conditions, it is important to further investigate the continuous flip-flop of the patch type, as the patches change behaviour from inlet to outlet and vice versa.

The space-time window reconstruction concept should be verified in other fields of science, like magnetohydrodynamics or electrotechnics.

For further validation, laser measurements could be performed in the space-time window region, in order to compare them against the results obtained by the reconstruction, at different levels of resolution.

References

[1] bwGRiD (http://www.bw-grid.de/), "Member of the German D-Grid initiative, funded by the Ministry of Education and Research (Bundesministerium für Bildung und Forschung) and the Ministry for Science, Research and Arts Baden-Wuerttemberg (Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg)," Universities of Baden-Württemberg, Tech. Rep., 2007-2010.

[2] J. Dongarra, H. S. Hans Meuer, and E. Strohmaier, "Biannual Top-500 computer lists track changing environments for scientific computing," http://www.top500.org/, 2000.

[3] E. Strohmaier and H. Meuer, "Supercomputing: What have we learned from the Top500 project ?" http://www.top500.org/, 2002.

[4] M. Ripeanu, "A note on the Zipf distribution of Top500 supercomputers," http://www.top500.org/, 2006.

[5] D. G. Feitelson, "On the interpretation of Top500 data," International Journal of High Performance Computing Applications, vol. 13, pp. 146–153, May 1999.

[6] S. S. Vazhkudai, X. Ma, V. W. Freeh, J. W. Strickland, N. Tammineedi, T. Simon, and S. L. Scott, "Constructing collaborative desktop storage caches for large scientific datasets," Trans. Storage, vol. 2, no. 3, pp. 221–254, 2006.

[7] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," IEEE Trans. Comput., vol. 58, no. 1, pp. 18–31, 2009.

[8] M. J. Quinn, Parallel Programming in C with MPI and OpenMP. McGraw-Hill Education Group, 2003.

[9] H. W. Meuer, "The TOP500 project: Looking back over 15 years of supercomputing experience," http://www.top500.org/, 2008.

[10] H. T. Kung, "Useful things to know about Ph.D. thesis research," Computer Science Department, Carnegie Mellon University, Tech. Rep., 1987.

[11] Anonymous, "History of computer developments in Romania," IEEE Annals of the History of Computing, vol. 21, no. 3, pp. 58–60, 1999.

[12] E. Strohmaier, "Top500 supercomputer," in Proceedings of the 2006 ACM/IEEE conference on Supercomputing, ser. SC '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1188455.1188474

[13] A.-A. Anton, "PhD Project Plan," "Politehnica" University of Timişoara, Tech. Rep. 3, 2009.

[14] R. D. Cook, Finite Element Modeling for Stress Analysis. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[15] N. Cormier, B. Smallwood, G. Sinclair, and M. G., "Aggressive submodelling of stress concentrations," International Journal for Numerical Methods in Engineering, vol. 46, no. 6, pp. 889–909, 1999.

[16] M. Burtscher and P. Ratanaworabhan, "gfpc: A self-tuning compression algorithm," in Proceedings of the 2010 Data Compression Conference, ser. DCC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 396–405. [Online]. Available: http://dx.doi.org/10.1109/DCC.2010.42

[17] J. Wilson, "Wavelet-based lossy compression of turbulence data," in DCC '00: Proceedings of the Conference on Data Compression. Washington, DC, USA: IEEE Computer Society, 2000, p. 578.

[18] K. Hiraki, M. Inaba, J. Tamatsukuri, R. Kurusu, Y. Ikuta, H. Koga, and A. Zinzaki, "Data reservoir: utilization of multi-gigabit backbone network for data-intensive research," in Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002, pp. 1–9.

[19] M. Kwan-Liu, W. Chaoli, Y. Hongfeng, and A. T., "In-situ processing and visualization for ultrascale simulations," Journal of Physics: Conference Series, vol. 78, no. 012043, 2007.

[20] H. P. Frits, V. Benjamin, H. Helwig, S. L. Robert, and D. Helmut, "The state of the art in flow visualisation: Feature extraction and tracking," 2003.

[21] A.-A. Anton, "CFD Application of Numerical Methods in Scientific Computing," Master's thesis, "Politehnica" University of Timişoara, 2009.

[22] B. Carlson, A. Burgess, C. Miller, and L. Bauer, "Timeline of computing history," Computer, vol. 29, pp. 1–34, October 1996. [Online]. Available: http://dl.acm.org/citation.cfm?id=244581.619527

[23] D. A. Grier, "Agricultural computing and the context for John Atanasoff," IEEE Annals of the History of Computing, vol. 22, no. 1, pp. 48–61, 2000.

[24] H. Goldstine and A. Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," IEEE Annals of the History of Computing, vol. 18, no. 1, pp. 10–16, 1996.

[25] T. H. Flowers, "The design of Colossus (foreword by Howard Campaigne)," IEEE Annals of the History of Computing, vol. 5, no. 3, pp. 239–252, 1983.

[26] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, "PETSc users manual," Argonne National Laboratory, Tech. Rep. ANL-95/11 - Revision 3.0.0, 2008.

[27] "OpenFOAM Programmers' Guide," http://www.openfoam.org/, July 2009.

[28] A.-A. Anton and V.-I. Creţu, "Unsupervised exploration of scientific articles," in Proceedings of the 5^{th} International Symposium on Applied Computational Intelligence and Informatics, Timişoara, May 2009, pp. 539–544.

[29] K. Leonard and R. Peter, "Clustering by means of medoids," International Conference on Statistical Data Analysis Based on the L1-Norm and Related Methods, August - September 1987.

[30] F. Murtagh and A. Heck, Multivariate Data Analysis with Astronomical Applications. Dordrecht, Holland: Kluwer Academic Publishers, 1987.

[31] Rousseuw, "Representing Data Partitions," Proceedings of the Statistical Computing Section of the American Statistical Association, pp. 275–280, 1985.

[32] A.-A. Anton, "A method for describing computational science," in The 3rd European DAAAM International Young Researchers' and Scientists' Conference, Vienna, November 2009, pp. 839–840.

[33] A. Sameh, G. Cybenko, M. Kalos, K. Neves, J. Rice, D. Sorensen, and F. Sullivan, "Computational science and engineering," Computing Surveys, vol. 28, no. 4, pp. 810–817, December 1996.

[34] G. Cybenko, "Computational culture shock," Computing in Science and Engineering, vol. 1, p. 1, 1999.

[35] A. Tveito and R. Winther, Introduction to Partial Differential Equations. A Computational Approach, 2nd ed. Springer-Verlag, 2009, vol. 29.

[36] G. P. Nikishkov, Programming Finite Elements in Java. Springer Publishing Company, Incorporated, 2010.

[37] A. B. Tucker, Computer Science Handbook, Second Edition. Chapman & Hall/CRC, 2004.

[38] A. D. Bader, Ed., Petascale Computing: Algorithms and Applications, ser. Computational Science. Chapman & Hall / CRC Press, Taylor and Francis Group, 2007.

[39] W. W. Benjamin, Ed., Wiley Encyclopedia of Computer Science and Engineering. John Wiley & Sons, Inc., February 2009.

[40] J. von Neumann and H. Goldstine, "Numerical inverting of matrices of high order," Bulletin of the AMS, vol. 53, no. 11, pp. 1021–1099, 1947.

[41] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3rd ed. Cambridge University Press, September 2007.

[42] R. B. Bird, W. E. Stewart, and E. N. Lightfoot, Transport Phenomena, 2nd ed. Wiley, Dec. 2006.

[43] R. Rojas, "How to make Zuse's Z3 a universal computer," IEEE Annals of the History of Computing, vol. 20, no. 3, pp. 51–54, 1998.

[44] ——, "Konrad Zuse's legacy: The architecture of the Z1 and Z3," IEEE Annals of the History of Computing, vol. 19, no. 2, pp. 5–16, 1997.

[45] J. Dongarra, "Performance of various computers using standard linear equations software in a fortran environment," ACM SIGARCH Computer Architecture News, vol. 11, pp. 22–27, December 1983.

[46] ——, "Performance of various computers using standard linear equations software in a fortran environment," ACM SIGNUM Newsletter, vol. 19, pp. 23–26, January 1984.

[47] ——, "Performance of various computers using standard linear equations software in a fortran environment," SIGARCH Comput. Archit. News, vol. 13, no. 1, pp. 3–11, 1985.

[48] ——, "Performance of various computers using standard linear equations software in a fortran environment," SIGARCH Comput. Archit. News, vol. 16, no. 1, pp. 47–69, 1988.

[49] ——, "Performance of various computers using standard linear equations software," SIGARCH Comput. Archit. News, vol. 20, no. 3, pp. 22–44, 1992.

[50] J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK benchmark: past, present and future," CONCURRENCY AND COMPUTATION-PRACTICE & EXPERIENCE, vol. 15, no. 9, pp. 803–820, AUG 2003.

[51] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The HPC Challenge (HPCC) benchmark suite," in SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing. New York, NY, USA: ACM, 2006, p. 213.

[52] A. Ruprecht, "Finite Elemente zur Berechnung drei-dimensionaler, turbulenter Strömungen in komplexen Geometrien," Ph.D. dissertation, Universität Stuttgart, 1989.

[53] D. P. Anderson, "BOINC: A system for public-resource computing and storage," in Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, ser. GRID '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 4–10. [Online]. Available: http://dx.doi.org/10.1109/GRID.2004.14

[54] D. G. Feitelson, "The supercomputer industry in light of the Top500 data," Computing in Science and Eng., vol. 7, no. 1, pp. 42–47, 2005.

[55] K. G. Zipf, Human behavior and the principle of least effort. Cambridge: Addison-Wesley Press, 1949.

[56] G. Bell, "Bell's law for the birth and death of computer classes: A theory of the computer's evolution," Microsoft Research, Tech. Rep., November 2007.

[57] S. Sharma, C.-H. Hsu, and W. chun Feng, "Making a case for a Green500 list." in IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)/ Workshop on High Performance - Power Aware Computing, 2006.

[58] W. Feng, M. Warren, and E. Weigle, "Honey, I shrunk the Beowulf!" in 2002 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, PROCEEDING, ser. PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, Abdelrahman, TS, Ed., Int Assoc Comp & Commun; Ohio State Univ; IBM Res; Microsoft Res; James Madison Univ, CISC. 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA: IEEE COMPUTER SOC, 2002, Proceedings Paper, pp. 141–148, 31st International Conference on Parallel Processing (ICPP 2002), VANCOUVER, CANADA, AUG 18-21, 2002.

[59] H. B. Newman, M. H. Ellisman, and J. A. Orcutt, "Data-intensive e-science frontier research," Commun. ACM, vol. 46, pp. 68–77, November 2003. [Online]. Available: http://doi.acm.org/10.1145/948383.948411

[60] M. Valero, "A european perspective on supercomputing," in ICS '09: Proceedings of the 23rd international conference on Supercomputing. New York, NY, USA: ACM, 2009, pp. 1–1.

[61] W. T. C. Kramer, A. Shoshani, D. A. Agarwal, B. R. Draney, G. Jin, G. F. Butler, and J. A. Hules, "Deep scientific computing requires deep data," IBM J. Res. Dev., vol. 48, no. 2, pp. 209–232, 2004.

[62] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," SIGMOD Rec., vol. 34, no. 4, pp. 34–41, 2005.

[63] A. Gerndt, T. V. Reimersdahl, T. Kuhlen, C. Bischof, I. Hörschler, M. Meinke, and W. Schröder, "Large-scale CFD data handling in a VR-based otorhinolaryngological CAS-system using a linux-cluster," J. Supercomput., vol. 25, pp. 143–154, June 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=778373.778387

[64] R. Burns, S. B. Davidson, Y. Ioannidis, M. Livny, and J. M. Patel, "Scientific data management: An orphan in the database community?" in ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. Washington, DC, USA: IEEE Computer Society, 2008, p. 9.

[65] K. Warendorf and A. Verhoeven, "HDF: a hypermedia document format for spatial hypermedia," J. Educ. Multimedia Hypermedia, vol. 8, no. 3, pp. 359–376, 1999.

[66] R. Rew and G. Davis, "Data management: NetCDF: an interface for scientific data access," IEEE Comput. Graph. Appl., vol. 10, no. 4, pp. 76–82, 1990.

[67] J. Li, W. keng Liao, A. Choudhary, R. Ross, R. Thakur, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel NetCDF: A high-performance scientific," in I/O Interface, Proc. Conf. Supercomputing, 2003.

[68] J. No, R. Thakur, and A. Choudhary, "Integrating parallel file I/O and database support for high-performance scientific data management," in Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM). Washington, DC, USA: IEEE Computer Society, 2000, p. 57.

[69] D. Korsemeyer and C. W. Thompson, "Guest editors' introduction: Internet access to scientific data," IEEE Internet Computing, vol. 9, no. 1, pp. 17–19, 2005.

[70] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)," United States, 2004.

[71] A. Chervenak, E. Deelman, C. Kesselman, B. Allcock, I. Foster, V. Nefedova, J. Lee, A. Sim, A. Shoshani, B. Drach, D. Williams, and D. Middleton, "High-performance remote access to climate simulation data: a challenge problem for data grid technologies," Parallel Comput., vol. 29, no. 10, pp. 1335–1356, 2003.

[72] A. Lucero, S. Cabrera, A. Aguirre, and E. Vidal, "Compressing three-dimensional GRIB meteorological data using KLT and JPEG 2000," in Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International, vol. 3, july 2003, pp. 1836 – 1838.

[73] J. L. Skidmore, M. J. Sottile, J. E. Cuny, and A. D. Malony, "A prototype notebook-based environment for computational tools," in Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM). Washington, DC, USA: IEEE Computer Society, 1998, pp. 1–15.

[74] V. P. Holmes, S. D. Kleban, D. J. Miller, C. Pavlakos, C. A. Poore, R. L. Vandewart, and C. P. Crowley, "An architecture and implementation to

support large-scale data access in scientific simulation environments," in SS '02: Proceedings of the 35th Annual Simulation Symposium. Washington, DC, USA: IEEE Computer Society, 2002, p. 169.

[75] H. M. Monti, A. R. Butt, and S. S. Vazhkudai, "Timely offloading of result-data in HPC centers," in ICS '08: Proceedings of the 22nd annual international conference on Supercomputing. New York, NY, USA: ACM, 2008, pp. 124–133.

[76] L. M. Kwan, "Visualization viewpoints: Visualizing visualisations," IEEE Computer Graphics and Applications, September/October 2000.

[77] D. Unat, T. H. III, and S. B. Baden, "An adaptive sub-sampling method for in-memory compression of scientific data," in Proceedings of the 2009 Data Compression Conference. Washington, DC, USA: IEEE Computer Society, 2009, pp. 262–271. [Online]. Available: http://dl.acm.org/citation.cfm?id=1545013.1545577

[78] H. Jasak, "Error analysis and estimation in the finite volume method with applications to fluid flows," Ph.D. dissertation, Imperial College, University of London, 1996.

[79] O. Rübel, Prabhat, K. Wu, H. Childs, J. Meredith, C. G. R. Geddes, E. Cormier-Michel, S. Ahern, G. H. Weber, P. Messmer, H. Hagen, B. Hamann, and E. W. Bethel, "High performance multivariate visual data exploration for extremely large data," in SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. Piscataway, NJ, USA: IEEE Press, 2008, pp. 1–12.

[80] R. R. Sinha and M. Winslett, "Multi-resolution bitmap indexes for scientific data," ACM Trans. Database Syst., vol. 32, no. 3, p. 16, 2007.

[81] L. M. Kwan and D. M. Camp, "High performance visualization of time-varying volume data over a wide-area network status," in Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM). Washington, DC, USA: IEEE Computer Society, 2000, p. 29.

[82] A. J. Smith, "Cache memories," ACM Comput. Surv., vol. 14, pp. 473–530, September 1982. [Online]. Available: http://doi.acm.org/10.1145/356887.356892

[83] A. Beszédes, R. Ferenc, T. Gyimóthy, A. Dolenc, and K. Karsisto, "Survey of code-size reduction methods," ACM Comput. Surv.,

vol. 35, pp. 223–267, September 2003. [Online]. Available: http://doi.acm.org/10.1145/937503.937504

[84] M. Thuresson, L. Spracklen, and P. Stenstrom, "Memory-link compression schemes: A value locality perspective," IEEE Transactions on Computers, vol. 57, pp. 916–927, 2008.

[85] M. A. Bassiouni, "Data compression in scientific and statistical databases," IEEE Trans. Softw. Eng., vol. 11, no. 10, pp. 1047–1058, 1985.

[86] F. Ghido, "An efficient algorithm for lossless compression of IEEE float audio," in Proceedings of the Conference on Data Compression. Washington, DC, USA: IEEE Computer Society, 2004, pp. 429–. [Online]. Available: http://dl.acm.org/citation.cfm?id=971935.972029

[87] X. Xie and Q. Qin, "Fast lossless compression of seismic floating-point data," in Proceedings of the 2009 International Forum on Information Technology and Applications - Volume 01. Washington, DC, USA: IEEE Computer Society, 2009, pp. 235–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1606748.1606807

[88] V. Engelson, D. Fritzson, and P. Fritzson, "Lossless compression of high-volume numerical data from simulations," in Proceedings of the Conference on Data Compression, ser. DCC '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 574–. [Online]. Available: http://dl.acm.org/citation.cfm?id=789087.789763

[89] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 5, pp. 1245–1250, 2006.

[90] P. Ratanaworabhan, J. Ke, and M. Burtscher, "Fast lossless compression of scientific floating-point data," in DCC '06: Proceedings of the Data Compression Conference. Washington, DC, USA: IEEE Computer Society, 2006, pp. 133–142.

[91] B. Goeman, H. Vandierendonck, and K. de Bosschere, "Differential FCM: Increasing value prediction accuracy by improving table usage efficiency," in Proceedings of the 7th International Symposium on High-Performance Computer Architecture, ser. HPCA '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 207–. [Online]. Available: http://dl.acm.org/citation.cfm?id=580550.876442

[92] M. Burtscher and P. Ratanaworabhan, "High throughput compression of double-precision floating-point data," in DCC '07: Proceedings of the 2007 Data Compression Conference. Washington, DC, USA: IEEE Computer Society, 2007, pp. 293–302.

[93] Y. Sazeides and J. E. Smith, "The predictability of data values," in Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture, ser. MICRO 30. Washington, DC, USA: IEEE Computer Society, 1997, pp. 248–258. [Online]. Available: http://dl.acm.org/citation.cfm?id=266800.266824

[94] V.-I. Creţu, Structuri de date şi algoritmi. Timişoara: Editura Orizonturi Universitare, 2000, vol. 1.

[95] M. Burtscher and P. Ratanaworabhan, "pFPC: A parallel compressor for floating-point data," in Proceedings of the 2009 Data Compression Conference. Washington, DC, USA: IEEE Computer Society, 2009, pp. 43–52. [Online]. Available: http://dl.acm.org/citation.cfm?id=1545013.1545555

[96] K. Sano, K. Katahira, and S. Yamamoto, "Segment-parallel predictor for FPGA-based hardware compressor and decompressor of floating-point data streams to enhance memory I/O bandwidth," in Proceedings of the 2010 Data Compression Conference, ser. DCC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 416–425. [Online]. Available: http://dx.doi.org/10.1109/DCC.2010.44

[97] H. Tomari, M. Inaba, and K. Hiraki, "Compressing floating-point number stream for numerical applications," in Proceedings of the 2010 First International Conference on Networking and Computing, ser. ICNC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 112–119. [Online]. Available: http://dx.doi.org/10.1109/IC-NC.2010.24

[98] M. A. O'Neil and M. Burtscher, "Floating-point data compression at 75 Gb/s on a GPU," in Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, ser. GPGPU-4. New York, NY, USA: ACM, 2011, pp. 7:1–7:7. [Online]. Available: http://doi.acm.org/10.1145/1964179.1964189

[99] J. Bradley, C. Brislawn, and T. Hopper, "FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression," Proc. of SPIE on Visual Information Processing II, vol. 1961,

no. 1, pp. 293–304, April 1993. [Online]. Available: http://dx.doi.org/doi/10.1117/12.150973

[100] K. Schneider and O. V. Vasilyev, "Wavelet Methods in Computational Fluid Dynamics," ANNUAL REVIEW OF FLUID MECHANICS, vol. 42, pp. 473–503, 2010.

[101] L. Kim, K. Nakahashi, H. Jeong, and M. Ha, "High-density mesh flow computations by building-cube method," Journal of Mechanical Science and Technology, vol. 21, pp. 1306–1319, 2007, 10.1007/BF03179047. [Online]. Available: http://dx.doi.org/10.1007/BF03179047

[102] H. Kang, D. Lee, and D. Lee, "A study on CFD data compression using hybrid supercompact wavelets," Journal of Mechanical Science and Technology, vol. 17, pp. 1784–1792, 2003, 10.1007/BF02983609. [Online]. Available: http://dx.doi.org/10.1007/BF02983609

[103] O. Christensen and K. Christensen, Approximation Theory: From Taylor Polynomials to Wavelets. Basel: Birkhäuser, 2004.

[104] M. Giles, "Wavelet compression for unsteady CFD data," Oxford University Computing Laboratory, Tech. Rep., October 1997.

[105] A. Trott, R. Moorhead, and J. McGinley, "The application of wavelets to lossless compression and progressive transmission of floating point data in 3D curvilinear grids," in Proceedings of the Conference on Data Compression. Washington, DC, USA: IEEE Computer Society, 1996, pp. 458–. [Online]. Available: http://dl.acm.org/citation.cfm?id=789084.789434

[106] J. Wilson, "Wavelet-based lossy compression of turbulence data," in Proceedings of the Conference on Data Compression, ser. DCC '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 578–. [Online]. Available: http://dl.acm.org/citation.cfm?id=789087.789766

[107] J. P. Wilson, "Wavelet-based lossy compression of barotropic turbulence simulation data," Data Compression Conference, vol. 0, p. 0479, 2002.

[108] K. Amaratunga, "A wavelet-based approach for compressing kernel data in large-scale simulations of 3D integral problems," Computing in Science and Eng., vol. 2, pp. 34–45, July 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=351387.351394

[109] R. Resiga, S. Muntean, S. Bernad, D. Balint, and I. Balint, Metode moderne de calcul paralel pentru simularea curgerii fluidelor. Editura Orizonturi Universitare, 2003.

[110] J. W. Liu, "A compact row storage scheme for Cholesky factors using elimination trees," ACM Trans. Math. Softw., vol. 12, pp. 127–148, June 1986. [Online]. Available: http://doi.acm.org/10.1145/6497.6499

[111] B. S. Andersen, J. A. Gunnels, F. G. Gustavson, J. K. Reid, and J. Waśniewski, "A fully portable high performance minimal storage hybrid format Cholesky algorithm," ACM Trans. Math. Softw., vol. 31, pp. 201–227, June 2005. [Online]. Available: http://doi.acm.org/10.1145/1067967.1067969

[112] B. P. Sommeijer and P. v. der Houwen, "Algorithm 621: Software with low storage requirements for two-dimensional, nonlinear, parabolic differential equations," ACM Trans. Math. Softw., vol. 10, pp. 378–396, December 1984. [Online]. Available: http://doi.acm.org/10.1145/2701.356103

[113] L. F. Shampine, "Storage reduction for Runge-Kutta codes," ACM Trans. Math. Softw., vol. 5, pp. 245–250, September 1979. [Online]. Available: http://doi.acm.org/10.1145/355841.355842

[114] Z. Bittnar, J. Kruis, J. Němeček, B. Patzák, and D. Rypl, "Parallel and distributed computations for structural mechanics: A review," in Civil and Structural Engineering Computing: 2001, B. Topping, Ed. Civil-Comp Press, 2001, pp. 211–233.

[115] A. E. H. Love, A Treatise on the Mathematical Theory of Elasticity. Cambridge: Cambridge University Press, 1927.

[116] E. P. N. Duque and S. M. Legensky, "Visualization of large-scale unsteady computational fluid dynamics datasets," in SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing. Washington, DC, USA: IEEE Computer Society, 2005, p. 73.

[117] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma, "In situ visualization for large-scale combustion simulations," IEEE Comput. Graph. Appl., vol. 30, pp. 45–57, May 2010. [Online]. Available: http://dx.doi.org/10.1109/MCG.2010.55

[118] A. Grama, G. Karypis, V. Kumar, and A. Gupta, Introduction to Parallel Computing (2nd Edition), 2nd ed. Addison Wesley, January 2003.

[119] S. Muntean, Analiza numerică a curgerii în turbinele hidraulice Francis. Timişoara: Editura Orizonturi Universitare, 2008.

[120] D. A. Lyn and W. Rodi, "The flapping shear layer formed by flow separation from the forward corner of a square cylinder," Journal of Fluid Mechanics, vol. 267, pp. 353–376, 1994.

[121] L.-E. Anton and A. Baya, Mecanica fluidelor, maşini hidraulice şi acţionări. Timişoara: Editura Orizonturi Universitare, 2002.

[122] A.-A. Anton, "A new method for reducing the storage requirements of numerical simulation data," in Proceedings of the 1^{st} IEEE ICCC-CONTI conference, Timişoara, May 2010, pp. 83–88.

[123] A.-A. Anton and V.-I. Creţu, "Accuracy of arbitrary subdomain reconstruction inside finite element simulations," in Proceedings of the 1^{st} IEEE ICCC-CONTI conference, Timişoara, May 2010, pp. 79–82.

[124] R. Resiga, Mecanica fluidelor numerică. Timişoara: Editura Orizonturi Universitare, 2003.

[125] J. R. Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, ser. Lecture Notes in Computer Science. Berlin: Springer-Verlag, May 1996, vol. 1148, pp. 203–222. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.9874

[126] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, and F. Booth, M.and Rossi, GNU Scientific Library Reference Manual - Third Edition. Network Theory Ltd., January 2009.

[127] R. Walter, Principles of Mathematical Analysis, 3rd ed. McGraw Hill, 1976.

[128] R. Pitz and J. Daily, "Combustion in a turbulent mixing layer formed at a rearward-facing step," AIAA Journal, vol. 21, pp. 1565–1570, Nov. 1983.

[129] IEEE, "IEEE standard for Floating-Point arithmetic," Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY 10016-5997, USA, Tech. Rep., Aug. 2008. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.2008.4610935

[130] A.-A. Anton, "Reconstruction of a space-time window in a transient simulation of the breaking of a dam," in Proceedings of the 6^{th} International Symposium on Applied Computational Intelligence and Informatics, Timişoara, May 2011, pp. 335–340.

[131] A.-A. Anton and V.-I. Creţu, "Fast algorithm for the extraction of a space-time window in finite volume meshes," in Proceedings of the 6^{th} International Symposium on Applied Computational Intelligence and Informatics, Timişoara, May 2011, pp. 327–330.

[132] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," ACM Comput. Surv., vol. 23, pp. 5–48, March 1991. [Online]. Available: http://doi.acm.org/10.1145/103162.103163

[133] F. P. Kärrholm, H. Weller, and N. Nordin, "Modelling injector flow including cavitation effects for diesel applications," ASME Conference Proceedings, vol. 2007, no. 42894, pp. 465–474, 2007. [Online]. Available: http://link.aip.org/link/abstract/ASMECP/v2007/i42894/p465/s1

[134] H. Simon, "Allen Newell: 1927-1992," Annals of the History of Computing, IEEE, vol. 20, no. 2, pp. 63 –76, Apr-Jun 1998.

Appendices

Thesis Research Activity and Publications

PhD Internship at the Institut für Strömungsmechanik und Hydraulische Strömungsmaschinen (IHS), University of Stuttgart, May-July 2010.

Supercomputer access through the bwGRID network and the facilities at the Höchstleistungsrechenzentrum Stuttgart (HLRS) , 2010 – 2011.

PhD Scholarship in the strategic grant POSDRU 6/1.5/S/13 2008, ID 6998.

1. Anton, A.-A. CFD Application of Numerical Methods in Scientific Comput- ing. Master's thesis, "Politehnica" University of Timişoara, 2009.

2. Anton, A.-A., and Creţu V.-I. Unsupervised exploration of scientific articles. In Proceedings of the 5^{th} International Symposium on Applied Computational Intelligence and Informatics (Timişoara, May 2009), pp. 539– 544.

3. Anton, A.-A. PhD Project Plan. Tech. Rep. 3, "Politehnica" University of Timişoara, September 2009.

4. Anton, A.-A. A method for describing computational science. In The 3^{rd} European DAAAM International Young Researchers' and Scientists' Conference (Vienna, November 2009), pp. 839–840.

5. Anton, A.-A., and Creţu, V.-I. Accuracy of arbitrary subdomain reconstruction inside finite element simulations. In Proceedings of the 1^{st} IEEE ICCC-CONTI conference (Timişoara, May 2010), pp. 79–82.

6. Anton, A.-A. A new method for reducing the storage requirements of numerical simulation data. In Proceedings of the 1^{st} IEEE ICCC-CONTI conference (Timişoara, May 2010), pp. 83–88.

7. Anton, A.-A., Reduced storage and bottleneck mitigation for the archival and offloading of parallel numerical simulation data, 6^{th} German-Romanian Workshop on Turbomachinery Hydrodynamics (GROWTH), 8–11^{th} of June, Stuttgart 2010

8. Anton, A.-A., and Vladimir-Ioan, C. Space-Time Window Reconstruction inside Finite Volume Simulations 5^{th} OpenFOAM Workshop, 21–24^{th} of June, Gothenburg 2010, Sweden

9. Anton, A.-A. A Method For Dealing with High Performance Numerical Simulation Results. Tech. Rep. 1, Universität Stuttgart, Institut für Strömungsmechanik und Hydraulische Strömungsmaschinen, July 2010.

10. Anton, A.-A. Reconstruction of a Space-Time Window inside Parallel High Performance Numerical Simulations by In-Situ Post-Processing of Approximated Interprocessor Traffic. Tech. Rep. 2, "Politehnica" University of Timişoara, September 2010.

11. Anton, A.-A. Reconstruction of a space-time window in a transient simulation of the breaking of a dam. In Proceedings of the 6^{th} International Symposium on Applied Computational Intelligence and Informatics (Timişoara, May 2011), pp. 335–340.

12. Anton, A.-A., and Creţu, V.-I. Fast algorithm for the extraction of a space- time window in finite volume meshes. In Proceedings of the 6^{th} International Symposium on Applied Computational Intelligence and Informatics (Timişoara, May 2011), pp. 327–330.

13. Anton, A.-A., Compression of Large Numerical Simulation Results, 7^{th} German-Romanian Workshop on Turbomachinery Hydrodynamics (GROWTH), 26–28^{th} of May, Timişoara 2011

14. Anton, A.-A. A Method for Submodelling Large Non-Stationary Flows. Tech. Rep. A, "Politehnica" University of Timişoara, July 2011.

15. Anton, A.-A., A method for submodelling inside transient flows, 13^{th} International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), September 26-29, Timişoara 2011

A book based on the PhD thesis has been published at Editura Politehnica Timişoara in 2011, having ISBN 978-606-554-390-4.